ZX SPECTRUM
ZX SPECTRUM +
QUÉ ES, PARA QUÉ SIRVE Y CÓMO SE USA



DR. TIM LANGDELL

EDITORIAL NORAY

ZX SPECTRUM

QUÉ ES, PARA QUÉ SIRVE Y CÓMO SE USA

ZX SPECTRUM

QUÉ ES, PARA QUÉ SIRVE Y CÓMO SE USA

DR. TIM LANGDELL

EDITORIAL NORAY
SAN GERVASIO DE CASSOLAS, 79
BARCELONA-22

Tim Langdell se ha ganado una merecida fama en el mundo de las computadoras a través de sus colaboraciones en revistas tales como Your Computer, Practical Computing y ZX Computing.

Recientemente, ha fundado una compañía de Software que tiene centrada su atención en el mundo de los micro ordenadores y en la creación de juegos imaginativos y aplicación de programas.

Para Cheri y Melissa y Sybil, Ted y Jenny Langdell y Sebastian.

Agradecimientos:

Vaya mi más sincero agradecimiento a todos los que han hecho posible esta publicación, especialmente a Richard Kennedy sin la ayuda del cual este libro no habría sido posible. También quiero darle las gracias a mi esposa Cheri por comprenderme y a Richard Gollner y Steven Adams por su inapreciable ayuda. Quiero agradecer especialmente a L.E. Listings del 1 Leswin Road, London N16 por toda su ayuda en los programas de este libro. Finalmente quiero agradecerle a Toby Wolpe que me facilitó el acceso a uno de los primeros Spectrums.

Título original: THE SPECTRUM HANDBOOK
Traducción de: RAMÓN ROVIRA

Tercera edición, 1984
Depósito Legal: B. 34972 - 1984
ISBN: 84-7486-037-7
Número de edición de E. N., 69
Printed in Spain - Impreso en España
GRÁFICAS INSTAR, S.A.
Industria, s/n - Hospitalet de Llobregat (Barcelona)

PROLOGO DEL TRADUCTOR

Este prólogo del traductor, obedece a que la mayoría de las personas que han adquirido un ZX-Spectrum, están impacientes por probarlo y no desean leerse el libro hasta después de haberlo visto funcionar. Por todo ello, he escrito estas instrucciones para el funcionamiento del casette, a fin de que se pueda utilizar desde el principio la cinta de demostración que acompaña al Spectrum o bien cualquier otra cinta que se haya adquirido.

Cuando habra la caja encontrará los siguientes objetos:

—El ordenador
—Un alimentador
—Un cable de antena
—Los cables del casette
—Manuales

Primero saque el ordenador y el alimentador de la caja, enchufe el alimentador a la corriente y el otro extremo a la entrada de la parte posterior del Spectrum que está marcada con el nombre "9V DC". Entonces el Spectrum emitirá un sonido muy débil y agudo.

Ahora, enchufe la televisión y seleccione un canal que esté en la banda UHF, conecte el cable de antena al televisor y el otro extremo a la entrada del Spectrum marcada "TV". Solo le queda sintonizar el canal escogido al número 36 del UHF. Si todo ha ido bien aparecerá la pantalla blanca con una K en la esquina inferior derecha.

Si esto no sucede asegúrese de que llega corriente al ordenador, esto se puede comprobar pulsando algunas teclas y escuchando si suena un "clic". Si ya ha comprobado que llega corriente al ordenador y en la pantalla sigue sin aparecer nada, compruebe la conexión del cable de antena, pues posiblemente esté mal conectado. Otra cosa que le puede suceder es que esté mal sintonizado, gire el botón de sintonía, poco a poco, asegurándose de que corresponde

a la banda del UHF. Si a pesar de todo ello sigue sin funcionar póngase en contacto con su distribudor y explíquele el problema, él sabrá cómo resolverlo.

Una vez tenga el Spectrum funcionando ya puede cargar la cinta de demostración. Para ello coja los cables del casette que vienen dentro de la caja, uno de ellos tiene las clavijas de color gris, y el otro las tiene negras. Ambos cables son iguales pero el color de las clavijas nos sirve para distinguir uno de otro. También necesitará un casette, que en principio puede ser cualquiera, pero ya sea por comodidad o por algún que otro detalle deberá tener unas características especiales, que son las siguientes:

Deberá disponer de una entrada para grabación (normalmente marcada como "MIC"), ya sea para grabar desde un micrófono exterior o bien desde otro aparato.

Deberá disponer de una salida. La más idónea es la dedicada a los auriculares, aunque también se puede utilizar la de un altavoz exterior. Esta salida está marcada normalmente como "EAR" o bien "External Speaker", en todo caso debe dar la suficiente potencia para que el sonido sea oído por el Spectrum.

Encontrará muy interesante que su casette disponga de contador de cinta pues es una gran ventaja a la hora de localizar los programas. Si no dispone de contador puede optar por decir los nombres de los programas por el micrófono antes de grabarlos, con lo cual facilitará su búsqueda.

También es interesante que disponga de nivel de volumen automático para la grabación. Así todos los programas que se graben lo harán al mismo volumen, evitando tener que cambiar de volumen a la hora de la carga. En caso de que no disponga de esta facilidad haga varias pruebas y grabe siempre al mismo volumen.

Para cargar un programa desde el casette al ordenador escoja uno de los dos cables (el gris o el negro) y enfuche un extremo a la salida "EAR" del casette y el otro a la entrada "EAR" del Spectrum. Después de esta operación rebobine la cinta hasta el principio del programa. Pulse la tecla "J" y aparecerá en la pantalla la palabra LOAD, seguidamente pulse la tecla "SIMBOL SHIFT" (que está en rojo en la parte inferior derecha del teclado) y manteniéndola pulsada, presione la tecla "P", esto hará que aparezcan unas comillas. Vuelva a pulsar la tecla "SIMBOL SHIFT" al mismo tiempo que la "P" para que aparezcan otras comillas, después de esta operación en la pantalla deberá aparecer:

LOAD" "

Una vez tenga esto en pantalla pulse la tecla "ENTER", entonces el borde de la pantalla irá cambiando de color. Es el momento de poner en marcha el casette en modo de reproducción (PLAY). Si le aparece un interrogante en algún lugar de la instrucción es que la ha escrito mal, bórrela y escríbala de nuevo.

Si todo ha ido bien, en cuanto el Spectrum detecte el programa aparecerán primero unas rayas rojas y azules en el borde durante unos segundos, para luego transformarse en rayas amarillas y azules más finas. Al realizar este cambio deberá aparecer el nombre del programa en la pantalla. Si no aparece, o bien obtiene el mensaje:

R Tape loading error, Ø:1

significa que algo ha ido mal. Repita el proceso, si en la pantalla no ha aparecido nada compruebe las conexiones, o suba el volumen del casette.

Una vez haya obtenido una carga correcta marque o fije el volumen empleado pues así no tendrá que hacer pruebas cada vez. Si dispone de control de tono o bien de graves y agudos, coloque siempre los agudos al máximo.

Tenga en cuenta que la instrucción:

LOAD" "

carga el primer programa que encuentra en la cinta pero si Ud. conoce el nombre del programa puede escribir:

LOAD "nombre"

y entonces el Spectrum dejará correr la cinta hasta que encuentre un programa con ese nombre. Pero cuidado, el nombre que escriba debe ser exactamente el nombre del programa incluso en lo que se refiere a las mayúsculas y a los espacios, así por ejemplo, el primer programa de la cinta de demostración se puede cargar rebobinando la cinta hasta el principio del programa y tecleando LOAD " " o bien haciendo:

LOAD "sidea"

seguido de ENTER pero no se cargará si UD. pone "Sidea" o "side a".

INTRODUCCIÓN

Si Vd. acaba de comprar su ZX-Spectrum y no conoce las técnicas de programación, entonces éste es el libro que Vd. necesita. Con él aprenderá los principios básicos de funcionamiento de su Spectrum, el sistema de lenguaje utilizado y cómo realizar sus propios programas (juegos, contabilidad casera e incluso aplicaciones comerciales).

Si Vd. ya conoce las computadoras pero se ha centrado en el ZX-Spectrum como ordenador personal, entonces la información contenida en este libro le será de gran valor. El Spectrum difiere de otros microordenadores en el tipo de BASIC utilizado, así como en el uso de *comandos de una sola tecla* que, habiendo sido diseñado para facilitar las cosas al principiante, puede confundir al entendido en otro tipo de micros.

A lo largo de todo el texto, obtendrá una visión general de cómo funciona el Spectrum, encontrará gran variedad de trucos que le identificará con la filosofía del Spectrum para que pueda explotarla al máximo en la máquina y verlos funcionar.

Este libro es aconsejable leerlo con su Spectrum delante, probando todo lo que se vaya explicando. Encontrará que las páginas contienen programas, juegos e información sobre la ZX-Spectrum y micros en general.

La primera sección le presenta a su nuevo Spectrum. No se asume que Vd. posea experiencia previa en ordenadores, cómo trabajan o qué se puede hacer con éstos. También he mantenido las matemáticas al mínimo, con lo cual se supone que Vd. podrá concentrarse en las técnicas de programación... Sin tener dolores de cabeza de segundo grado. Cualquiera que sea el nivel de conocimiento de las técnicas de programación que Vd. obtenga mediante la lectura de este libro, seguro que quedará convencido que

la programación de su ZX-Spectrum puede ser francamente divertida. Ante todo, si queda bloqueado en algún punto del libro, tómese un descanso y vuelva al mismo punto más tarde. No se amargue por nada, recuerde que tiene que ser una experiencia agradable.

Hacia el final de la primera sección Vd. tendrá ya bastante buen conocimiento de las posibilidades del Spectrum y de los fundamentos para escribir programas bien estructurados. Pruebe todos los programas y ejemplos, con lo cual cada vez le serán más fáciles los siguientes pasos a dar.

En las restantes secciones, se estudia cada una de las características importantes del Spectrum y se examinan sus posibilidades a fondo. Vd. aprenderá a crear figuras sorprendentes en su pantalla de TV, escribir juegos con gráficos en movimiento, crear sus propios intergalácticos, tocar música, escribir programas útiles para casa y para la escuela... y mucho, mucho más.

¡Espero que disfrute de la lectura de este libro tanto como yo disfruté escribiéndolo!

Tim Langdell

Primera Parte
Introducción al ZX Spectrum

CAPÍTULO PRIMERO
Éste es su ZX Spectrum

Una nota para el lector más experimentado.
Si usted ha usado ordenadores anteriormente, la mayoría de las cosas que se cuentan en esta primera parte le serán familiares, pero dado que en ella se explican las particularidades y posibilidades del Spectrum, sugiero que la lea por encima para descubrir así las habilidades de su nueva máquina.

CÓMO TRABAJA EL ORDENADOR

El ZX Spectrum puede muy bien ser el primer ordenador que usted haya visto jamás (al menos en persona), y puede que no sepa mucho sobre ordenadores, lo que pueden hacer, etc., ...así, que empezaremos con la información básica necesaria.

Todos los ordenadores tienen en su interior un componente electrónico llamado **microprocesador,** que puede pensarse como el cerebro de la máquina. Esta pieza no es muy grande, parece un bloque de plástico negro con cerca de cuarenta patillas que le salen por debajo. En realidad, el microprocesador propiamente dicho (también llamado **unidad central de proceso** o **CPU)** se encuentra en el interior del plástico y es muy pequeño: más o menos como la mitad de la uña del dedo meñique.

Este microprocesador hace actualmente lo mismo que una habitación llena de válvulas hiciera hace sólo unos pocos años. No vamos a entrar aquí en la complejidad de la electrónica, pues nos ocuparía otro libro entero, pero voy a dar una pequeña idea de lo que es. Esencialmente, la CPU, "piensa" en términos de voltajes que pueden ser altos (encendido) o bajos (apagado). Estos dos estados pueden ser pensados como unos y ceros. El ordenador recibe grupos de ocho de estos ceros y unos, y los trata al igual que haríamos nosotros con una palabra. Cada grupo de éstos se llama un **byte** o también una **palabra,** y cada

uno o cero de un byte se llama **bit.** Por lo tanto, un byte se compone de ocho bits.

Ahora bien, el lenguaje de bits y bytes que entiende el CPU del Spectrum es muy complicado. Es por eso que para poder hablar con el ordenador en una forma mucho más simple se inventó un lenguaje especial que se parece al inglés. Este lenguaje se llama **BASIC.** (Beginners All-purpose Symbolic Instruction Code). La CPU del ZX Spectrum no entiende BASIC, por eso hay otro componente electrónico, la **ROM (memoria de sólo lectura)** que traduce las palabras del BASIC parecidas al Inglés al lenguaje de ceros y unos. El problema está en que para conseguir este milagro el Spectrum es muy exigente a la hora de admitir las palabras que le permitirán conversar con él. Usted y yo podemos decir la misma cosa de distintas maneras y entendernos perfectamente aunque usemos distintas palabras. Por ejemplo, una de las palabras más importantes del BASIC del Spectrum es **PRINT.** Para un inglés es fácil predecir que esta instrucción servirá para que el Spectrum imprima alguna cosa en la pantalla de TV. Pero en cambio, mientras que usted y yo entenderíamos también "write", "type", o "display", que son palabras inglesas que significan lo mismo que "print", su Spectrum no puede. Para ayudarle a recordar qué palabras pueden decírsele al Spectrum éste dispone de un sistema muy bien pensado para la introducción de este número limitado de palabras clave, que *no* tienen otros ordenadores. Es un sistema de entrada mediante una sola tecla. Es decir, que la primera tecla que usted pulse no imprimirá una sola letra en la pantalla (como hace una máquina de escribir en el papel) sino que aparecerá una palabra entera. Así por ejemplo, apretando la tecla "P" aparecerá la palabra PRINT. Si usted se olvida de esto e intenta escribir "write" en lugar de PRINT en seguida notará su error, ya que en vez de WRITE le aparecerá:

DRAW rite

Antes de empezar a explorar el Spectrum permítame decirle que el mundo de los ordenadores tiene su propia jerga; extrañas palabras como bits, bytes, CPU, floppy disk, enter, etc. A medida que veamos las posibilidades del Spectrum, iré introduciendo muchas de estas palabras. Por lo

pronto hay una en particular que le debe haber preocupado, es el uso de la letra "K", por ejemplo: "El Spectrum de 16 K", "Spectrum de 48 K"... Uno es más de diez mil pesetas más caro que el otro. "K" significa mil bytes, y es una indicación de qué cantidad de información es capaz de almacenar un ordenador. En realidad un "K" no son mil bytes sino 1024, pero es más cómodo hablar de mil.

Sin entrar en el aspecto técnico, se puede decir que el Spectrum de 16K tiene aproximadamente 9K de memoria utilizable, mientras que el de 48K tiene alrededor de 41. Se llama memoria al espacio en el cual se pueden escribir programas. Por lo tanto, se puede decir que el de 48K tiene cuatro veces más de memoria utilizable que el de 16K.

CONEXIÓN

Habiendo enchufado su Spectrum tanto a la corriente como a la televisión, obtendrá en la parte baja de la pantalla un mensaje como este:

© 1982 Sinclair Research Ltd

Pulse la tecla **ENTER** que está situada en lado derecho del teclado. El mensaje desaparece y una "K" parpadeante aparece en la esquina inferior izquierda de la pantalla. Esta "K" es llamada **cursor**, y es uno de los seis indicadores de que dispone el Spectrum. El cursor le indica en qué parte de la pantalla está, es decir, dónde aparecerá lo próximo que se imprima. Pero el cursor, también indica qué es lo que el Spectrum está esperando. Esto es exclusivo de las máquinas Sinclair ya que la mayoría de ordenadores disponen únicamente de un solo cursor (representado normalmente por ">").

El cursor K le indica a usted que el Spectrum está esperando una *palabra clave* (en inglés, Keyword). Si ahora pulsa una tecla observará que en lugar de aparecer la letra correspondiente va a aparecer una palabra entera. Por ejemplo, cuando el cursor está en modo K, si usted pulsa la tecla P aparecerá en pantalla la palabra PRINT. Acto seguido, el cursor cambia a modo L lo cual significa que el Spectrum está esperando el resto del mensaje. Pulsando cualquier tecla con el cursor en modo L, lo que aparece es

el número o la letra correspondiente a esta tecla, más o menos lo que ocurriría si usted estuviera usando una máquina de escribir. Ahora puede hacer dos cosas o bien usar la tecla **CAPS SHIFT** junto con 0 para borrar lo que ha escrito o bien pulsar ENTER. Si se ha decidido por **DELETE,** espero que notará que las teclas disponen de autorrepetición. Esto significa que al cabo de un momento de haber pulsado una tecla ésta se repite sin necesidad de que usted vuelva a pulsarla. Por lo tanto, para borrar una línea muy larga sólo tiene que pulsar CAPS SHIFT y 0 simultáneamente, esperar a que se borre la primera letra y mantener el dedo en el 0 mientras se borra la línea rápidamente.

CONSEJO

> Para deshacerse rápidamente de todo lo que haya escrito en la parte inferior de la pantalla pulse CAPS SHIFT y la tecla "1", esto produce la edición de una línea de programa; pero si no hay ningún programa, entonces se limpia toda la parte baja de la pantalla. Si hubiera algún programa entonces "bajaría" una línea, eliminando lo que había anteriormente, luego pulse ENTER para devolverla arriba.

Si en cambio usted pulsó ENTER, entonces, obtendrá un mensaje de este tipo:

0 OK, 0:1

Esto significa "Ya lo he hecho, ¿qué más", aunque también puede ser que tenga menos suerte y le aparezca un **mensaje de error** (como: "2 Variable not found") o bien una S parpadeante. El mensaje de error es el modo que tiene el Spectrum de decir que no entiende lo que hemos escrito. La otra respuesta (la S) tiene un significado similar, sólo que además nos indica dónde se ha producido el error. La S significa error de Sintaxis, pero volveremos a ella más tarde, de momento tendrá que usar DELETE o bien el consejo anterior para continuar.

Otro cursor que nos encontraremos es el cursor E. También parpadea y significa "Extended-mode", volveremos a

él en un momento, por ahora sólo observe que aparece pulsando CAPS SHIFT y SIMBOL SHIFT simultáneamente. Existe todavía otro cursor en el Spectrum, es el cursor C. Éste nos indica que las letras que se pulsan serán mayúsculas ("Capitals" en inglés) en lugar de minúsculas. Este cursor aparece pulsando CAPS SHIFT y la tecla "2" (observe que pone CAPS LOCK encima de la tecla), y vuelva a la forma L actuando del mismo modo. De esto se deduce que la L significa minúsculas (Lower case en inglés). El último cursor es el cursor G que indica que nos encontramos en modo Gráfico. Se obtiene pulsando CAPS SHIFT y "9".

Ahora vamos a explorar el teclado en general, pero primero remarcaré que acabo de usar Ø para indicar el número cero. Esto es una práctica corriente en ordenadores para distinguir claramente el Ø de la letra O.

Observe que esto mismo ocurre en la pantalla del Spectrum, así, como en el teclado. El Spectrum también dispone de color, para obtenerlo usted debe saber cómo decirle a la máquina que lo produzca. Este tema lo trataremos más adelante, pero de momento, para hacer la pantalla más bonita escriba lo siguiente:

BORDER 3 (y pulse ENTER, la palabra BORDER se obtiene pulsando la tecla B con el cursor en modo K, procure escribirlo exactamente sino le producirá un error).

EL TECLADO

A primera vista, el teclado del Spectrum parece un poco confuso con sus 191 palabras, caracteres y símbolos tan sólo en 4Ø teclas, pero en realidad es muy fácil de usar y usted lo comprobará muy pronto por sí mismo.

Empezaremos por ver cómo se obtienen los distintos símbolos de cada tecla. Las reglas son muy simples. Ya habrá notado que pulsando cualquier tecla cuando el cursor está en modo K, lo que se obtiene es la palabra en blanco que está escrita sobre la tecla.

Para obtener cualquiera de las palabras o abreviaciones que están en verde encima de las teclas, tiene que pulsar CAPS SHIFT y SYMBOL SHIFT simultáneamente para

que el cursor pase a modo E, después de esto, pulsando cualquier tecla, lo que aparecerá será la palabra en verde correspondiente a esa tecla. Para obtener las palabras que están en rojo, debajo de las teclas, se procede: primero a colocar el cursor en modo E tal como acabo de explicar, y después se pulsa la tecla al mismo tiempo que CAPS SHIFT o SYMBOL SHIFT.

Luego están los símbolos en rojo que se encuentran en las propias teclas. Estos se obtienen pulsando SYMBOL SHIFT al mismo tiempo que la tecla. No se confunda, pues esta operación también produce algunas palabras como AND, OR o AT, pero su Spectrum es muy coherente y si usted recuerda que SYMBOL SHIFT sirve para obtener las palabras en rojo estará en lo correcto.

Luego están los caracteres en blanco en la fila de arriba. Hemos visto ya cómo obtener CAPS LOCK, y usted recordará que había que pulsar CAPS SHIFT al mismo tiempo que la tecla 2. Pues bien, las otras palabras en blanco se obtienen de la misma manera. CAPS LOCK significa que todas las teclas que se pulsen producirán letras mayúsculas, exceptuando las de la fila de arriba que producen la palabra o símbolo que está en blanco debajo del color si lo hay. Las cuatro flechas se llaman: flechas del cursor y permiten mover a éste a lo largo de la pantalla.

Por último, están los símbolos gráficos que se obtienen pulsando CAPS SHIFT al mismo tiempo que la tecla 9. Esto produce la aparición del cursor G y ahora, cuando usted pulse cualquiera de las teclas del 1 al 8 obtendrá los caracteres gráficos. Si usted pulsa alguna de las otras teclas observará que produce una letra, que podrán ser reemplazadas por sus propios caracteres gráficos (como hombres, piedras, alfabeto griego, etc.), pero ya veremos más adelante cómo hacerlo. Pulsando una tecla desde la A hasta la U aparece una letra mayúscula, pero las teclas desde la V hasta la Z producen RND, INKEYS, PI, FN y POINT respectivamente. Usted no necesita saber lo que esto significa ahora, ni siquiera recordar cómo se obtienen, puesto que se puede hacer de un modo más normal.

De todos modos intente evitar el pulsar las teclas de la fila de arriba, ya sea con o sin SHIFT, cuando tenga el cursor en modo E ya que esto produce unos efectos extraños que ya explicaremos en su momento.

DOS COSAS A RECORDAR

Para salir del modo G o C se repite lo mismo que para entrar. El modo E sólo sirve para una pulsación, pero si quiere salir de él sin efectuar ninguna, repita lo mismo que para entrar. También recuerde que si usted quiere varias veces los mismos símbolos, cada tecla se repite automáticamente. Pero cuando se usa algún símbolo que necesita de las teclas SHIFT no es necesario mantener el dedo en ella para que se efectúe la repetición.

PALABRAS CLAVE

Ahora que ya ha visto cómo obtener impresos en la pantalla todos los símbolos que dispone el Spectrum, profundizaremos un poco en algunas de las palabras clave que se producen cuando el cursor está en modo K.

Intente pulsar alguna tecla, observará que la primera vez se imprime una palabra completa mientras que la próxima tecla imprime una letra, un número o un símbolo. Una vez escritas varias palabras o caracteres pulse la tecla ENTER, probablemente obtendrá el cursor S al principio de la línea. El cursor S indica error de sintaxis, que básicamente significa que su Spectrum cree que usted quería introducirle un comando que él no pudo entender. El cursor S está situado exactamente donde el Spectrum piensa que usted cometió el error.

ESCRITURA EN LA PANTALLA

¿Qué comando espera el Spectrum? Como hemos visto, uno de los más usados es PRINT, que se obtiene pulsando la tecla P cuando el cursor está en modo K. El uso de este comando es muy sencillo, pero hay algunas reglas muy simples que deben quedar claras. Intente escribir exactamente lo mismo que se indica más abajo, encontrará las comillas en la tecla P y la coma en la N apretando para ambos SYMBOL SHIFT, como ya he explicado.

```
PRINT "hola, yo soy su zx spectrum"
```

Ahora pulse la tecla ENTER y verá que en la parte superior de la pantalla aparece:

hola, yo soy su zx espectrum

y en la parte inferior de la pantalla aparecerá un mensaje que dice:

0 OK, 0:1

recordemos que este mensaje significa: "he terminado, ¿alguna cosa más?". Por supuesto que podríamos haber usado letras mayúsculas en la frase. Ahora intente obtener este mensaje por la pantalla recordando que debe usar CAPS SHIFT para obtener las mayúsculas:

¿Qué puedo hacer por usted?

¿SON LAS COMILLAS REALMENTE NECESARIAS?

Habrá observado que para imprimir cualquier cosa en la pantalla, se usa el comando PRINT seguido de la frase que se quiere imprimir encerrada entre comillas. Pero ¿son necesarias las comillas? Intente escribir:

PRINT hola

ahora pulse ENTER. "Hola" no aparece por ningún lado, en su lugar se obtiene el siguiente mensaje de error:

2 Variable not found, 0:1

El problema está en que le hemos pedido al Spectrum que imprima una cosa que *no* tiene comillas, entonces él piensa que se trata de un número y no puede imprimir "hola" como si fuera un número. De esto último se deduce que los números pueden ser impresos en pantalla directamente. Para probar esto escriba algunos números de la siguiente manera:

PRINT 2 (y pulse ENTER)
PRINT 85 (y pulse ENTER)

El Spectrum también sabe matemáticas, pruebe algunas operaciones, teniendo en cuenta que "*" se usa para la

multiplicación, "/" para la división y "↑" para las potencias:

```
PRINT 2 + 4          (pulse ENTER)
PRINT 34/97          (pulse ENTER)
PRINT 2 ↑ 2          (pulse ENTER)
PRINT 3*54/9         (pulse ENTER)
```

Con esto habrá obtenido los siguientes números en la parte superior de la pantalla: 6,0.35051546, 4,18.

Así que su Spectrum puede ser usado como una calculadora, aunque un poco cara, pero si quiere utilizarlo como tal será bueno que sepa que el Spectrum trata las diferentes operaciones matemáticas con prioridades distintas, en otras palabras, en una expresión muy larga, todas las sumas se efectúan después de los productos. La siguiente tabla nos muestra las distintas prioridades que concede el Spectrum a las diferentes operaciones matemáticas:

Operación	Prioridad
↑	10
- (negativo)	9
*	8
/	8
+	6
- (resta)	6

La lista completa de las prioridades va desde el 16 hasta el 1. Los paréntesis tienen la más alta, pasando por las funciones, operaciones lógicas, y AND, OR y NOT que tienen la más baja. De todas maneras, la regla de oro es usar los paréntesis siempre que se tenga alguna duda, ya que lo es está dentro de los paréntesis se calcula siempre antes que lo que está fuera. Por ejemplo, la siguiente expresión sin paréntesis puede valer 104 o bien 102:

```
PRINT 2*50 + 2
```

Usted puede mirar a la tabla anterior para saber lo que hará su Spectrum, pero usando los paréntesis en alguna de estas dos formas no habrá ninguna duda:

PRINT 2*(5Ø + 2)   o bien   PRINT (2*5Ø) + 2

Hay que notar que no se usa el signo de multiplicar "×" porque se parece demasiado a la letra X. En su lugar se usa "*", del mismo modo que se usa "/" en vez de "÷" para la división.

Finalmente, el Spectrum puede tratar decimales y no únicamente números enteros:

PRINT 7,34*9,672

Resumiendo, si usted está imprimiendo un número no necesita usar comillas; en cambio si quiere imprimir una frase, una palabra o cualquier otra cosa que deba aparecer en pantalla "tal cual" debe encerrarse entre comillas. Para ver la diferencia escriba:

PRINT 2 + 2   (y pulse ENTER)
PRINT "2 + 2"   (y pulse ENTER)

BORRADO DE PANTALLA

Si ha ido probando los ejemplos es probable que tenga ahora la pantalla llena de cosas, para borrarla utilice el comando **CLS**, que se obtiene pulsando la tecla V. Una vez hecho esto, le aparecerá el ya familiar informe: Ø OK, Ø:1.

El Spectrum +

A finales de 1984 Sinclair anunció el lanzamiento del «Spectrum +»

El Spectrum + no es más que el conocido Spectrum 48 K con un nuevo diseño externo que incluye un «verdadero» teclado. El diseño del Spectrum + está inspirado en el del nuevo Sinclair QL, también lanzado en 1984. Realmente el Spectrum + y el QL son muy parecidos.

EL NUEVO TECLADO

La principal diferencia entre el ZX Spectrum de 48 K y el Spectrum + está en el teclado y en la presencia en el Spectrum + de algunas teclas nuevas. Las teclas del Spectrum + son más parecidas a las que podemos encontrar en una máquina de escribir, pero una vez reconocido que son

mejores que las de los otros Spectrums, también añadiremos que sigue siendo difícil escribir con rapidez. El Spectrum + está dotado de una «barra espaciadora», pero las mejoras más importantes se centran en la introducción de ciertas teclas que realizan funciones que en los otros Spectrums exigían apretar varias teclas. Estas teclas nuevas son: DELETE, GRAPH, EXTEND MODE, EDIT, CAPSLOCK, BREAK, TRUE VIDEO e INV VIDEO. También tienen teclas independientes los signos: punto y coma, comillas y coma. Las teclas de movimiento del cursor tambien están separadas, dos a cada lado de la barra espaciadora que, tal como cabe esperar, está en el centro de la fila inferior del teclado.
La presencia de las nuevas teclas simplifica en muchos casos la programación.

Aunque para mover el cursor por la pantalla puede usarse el SHIFT y la tecla del número correspondiente, en el Spectrum + es más fácil emplear las teclas del cursor, claramente marcadas con flechas. Cambiar al cursor E «Extend Mode» también es más fácil, así como borrar los errores. Todo esto puede hacerse ahora con una sola tecla.

OTROS CAMBIOS

Excluyendo el evidente aumento de tamaño del Spectrum + (es cerca de una vez y media mayor que el otro), tiene relativamente pocos cambios. Sinclair nos asegura que los programas para Spectrum pueden correr sin necesidad de adaptación en el Spectrum +.

Las entradas y conexiones de la parte posterior son iguales en las dos versiones, lo que permite pensar que todo el hardware existente para el Spectrum puede utilizarse con el Spectrum +. De todas formas el Spectrum + tiene unos pequeños apoyos que pueden extraerse de su parte posterior permitiendo que el aparato quede inclinado hacia adelante. Algunos equipos complementarios del Spectrum se apoyan sobre la superficie de la mesa y no trabajarán bien si están proyectados a media altura en la parte posterior del ordenador. Los únicos cambios restantes son la inclusión de un botón «reset» y una cierta mejora en el apartado del sonido. El botón «reset» está bien escondido en la parte izquierda del Spectrum +, para evitar que sea pulsado accidentalmente, bastante a la izquierda de la tecla DELETE. Pulsar «reset» tiene el mismo efecto que desconectar y volver a conectar el Spectrum de la fuente de alimentación, al realizar esto electrónicamente el Spectrum

+ trata mejor a sus componentes que el Spectrum normal (muchos propietarios de estos aparatos, a menos que el teclado haya quedado sin efecto, prefieren emplear RANDOMISE USR Ø, antes que desenchufar y volver a enchufar). Recuerde pues que apretar este botón produce el mismo efecto que el comando NEW, ¡desaparecerá todo lo que el ordenador tenga en la memoria!

Por último, podemos resaltar que es más difícil leer el teclado del Spectrum + que el de los Spectrum anteriores. En el Spectrum cada tecla contiene un signo en color rojo, otros en blanco, además de los signos verde y rojo impresos en la parte superior e inferior contigua a cada tecla. En el teclado del Spectrum + no se utilizan colores, las teclas tienen todos los símbolos en blanco. La ausencia de los códigos de color hace que sea más complicado el uso de los distintos «modos». Por ejemplo, se hace más difícil saber qué símbolo o tecla debe apretarse si nos encontramos en «extend mode» dependiendo de cuando hemos de apretar la tecla SHIFT o no.

De todas formas, con un poco de práctica se descubre que el sistema del Spectrum + es muy lógico, e incluso llega a ser más práctico que el de los otros Spectrum. Cada tecla, al ser pulsada con el cursor en «K» producirá la palabra clave que aparece en la parte central más alta de la misma tecla. Si se está apretando SYMBOL - SHIFT al mismo tiempo, se producirá la palabra o símbolo situado justo encima de la letra o número principal. Cuando se trabaja en «extend mode» se produce la palabra situada en la parte superior del cuadrado de cada tecla. Si estando en «extend mode» se pulsa SYMBOL SHIFT simultáneamente a la tecla deseada, se produce la segunda palabra que aparece en el cuadrado. Afortunadamente cuando se tiene un poco de práctica, utilizar el teclado es mucho más fácil de lo que pueda parecer al leer esta explicación.

Una puntualización sobre la «compatibilidad» del software. Cuando Sinclair afirma que todo el software producido para el Spectrum de 48K corre perfectamente en el Spectrum +, no hay que tomarlo al pie de la letra. En algunos programas (juegos especialmente) el programador ha seleccionado algunas teclas de control que pueden no ser las más adecuadas en el Spectrum +, donde algunas teclas están colocadas en sitios diferentes. De todas formas, no creo que esto pueda ser un gran obstáculo para disfrutar con todo el software que no haya sido diseñado pensando en el nuevo modelo.

CAPÍTULO SEGUNDO

Color y sonido

LA ANIMACIÓN DEL COLOR

El Spectrum es un ordenador con color. La manera más fácil de obtenerlo es utilizando la instrucción **BORDER.** Este comando del BASIC, lo puede obtener pulsando la tecla B. BORDER debe ir seguido de un número entre 0 y 7 que le dice al Spectrum de qué color queremos el borde de la pantalla. Introduzca lo siguiente:

BORDER 1

El borde de la pantalla se volverá azul, a no ser que haya olvidado algo (compruebe que ha pulsado exactamente la tecla B y que después ha pulsado ENTER). Pruebe con esto usando otros números que estén entre 0 y 7. El color que representa cada número está escrito (en ese color) encima de las teclas que van del 0 al 7. Observe que el azul claro se llama CYAN y el púrpura, MAGENTA. Para mayor comodidad aquí tiene una tabla de los colores con sus números, a los cuales se les llama **código** del color.

0 NEGRO
1 AZUL OSCURO
2 ROJO
3 MAGENTA
4 VERDE
5 CYAN
6 AMARILLO
7 BLANCO

De hecho, este orden de colores tiene una causa que se ve enseguida si usted tiene un televisor en blanco y negro o bien le quita el color completamente. Una vez hecho esto, la representación de los colores del cero al siete se ve con diferentes estados del gris que van desde el más oscuro que es el negro (0) hasta el más claro, el blanco (7). Así por ejemplo, el magenta es más claro que el rojo, el verde más que el magenta, etc.

Bien. Pero ¿qué pasa con el color en el resto de la pantalla? El rectángulo que está dentro del borde se llama **PAPER** (papel). Así que para colorearlo debe usar el comando PAPER. Para obtener esta instrucción debe pulsar tres teclas consecutivamente. Primero CAPS SHIFT y SYMBOL SHIFT simultáneamente, entonces el cursor pasa a modo E. Luego, pulsando CAPS SHIFT o SYMBOL SHIFT apriete la tecla C y la palabra PAPER aparecerá en la pantalla. PAPER debe ir seguido de un número al igual que BORDER para indicar el color. Este número debe estar entre 0 y 7, y representan los mismos colores. Con las siguientes instrucciones se obtiene el borde azul oscuro y el papel azul claro (cyan).

```
BORDER 1

PAPER 5 (PAPER debe ir seguido de CLS)
CLS
```

Usted también puede cambiar el color de los números y letras que aparecen en pantalla. Esto se hace usando la instrucción **INK.** Para obtenerla se debe colocar el cursor en modo E igual que antes y después pulsar la tecla "X" junto con cualquiera de las teclas SHIFT. Al igual que BORDER y PAPER, INK debe ir seguido de un número en las mismas condiciones. Lo que sigue ahora es un ejemplo en el que se escribe una palabra en amarillo sobre fondo verde, intente transcribirlo exactamente, incluyendo los signos de puntuación. (El punto y coma se obtiene pulsando SYMBOL SHIFT y O):

```
PRINT PAPER 4; INK 6; "Hola"
```

Como puede ver, el color puede ser cambiado tanto localmente como en toda la pantalla.

Por último, usted puede cambiar la tinta (INK en inglés) de todo lo que hay escrito en la pantalla. Al conectar el Spectrum la tinta es negra, para cambiarla sólo tiene que escribir INK seguido del número correspondiente (y pulsar ENTER por supuesto) y luego CLS. Pruebe con esto:

```
PAPER 1
CLS
INK 5
CLS
```

Ahora podrá escribir palabras en azul claro sobre papel azul oscuro. Compruébelo escribiendo distintos comandos PRINT, pero no incluya en ellos ninguna instrucción INK o PAPER. Por ejemplo, esto le proporcionará la palabra Spectrum escrita en azul claro:

```
PRINT "Spectrum"
```

BRILLO Y PARPADEO

Existen tres comandos más que pueden afectar a los colores de la pantalla y que al igual que los otros se encuentra en la última fila de teclas del Spectrum. Estas palabras son **BRIGHT** (Brillo), **FLASH** e **INVERSE**.

BRIGHT hace que un color sea más o menos brillante. Se puede obtener al igual que los otros comandos de color que ya hemos visto: Primero se pone el cursor en modo E, y luego pulsando una de las teclas SHIFT se pulsa la B. Al contrario que los otros comandos de color, BRIGHT no va seguido de un código de color sino que se pone en 1 si quiere brillo o bien un Ø si no se quiere. Por ejemplo, escriba lo siguiente:

```
PRINT BRIGHT 1; PAPER 4; INK Ø; "Hola"
```

y compárelo con esto otro:

```
PRINT BRIGHT Ø; PAPER 4; INK Ø; "Hola"
```

La segunda instrucción tiene el papel más oscuro que la primera. Y es lo mismo escribir BRIGHT Ø que no poner BRIGHT. Pero existe una razón para que haya esta posibi-

lidad de poner 0 y 1 después de BRIGHT. Cuando se escribe un programa, asunto este que trataremos más tarde, usted puede querer escribir una cosa brillante en un momento determinado y en color normal en otro momento. Por ejemplo usted quizá quiere indicar la posición de un jugador en un juego en brillante, y la del otro no. En lugar de utilizar dos instrucciones distintas, una con BRIGHT y la otra sin, se puede hacer en una línea sola. Mire esto:

PRINT BRIGHT X; PAPER 2: "posición"

Aquí X se usa como una "variable" (no se preocupe ahora por lo que significa esta palabra). Haciendo X igual a 0 o a 1 usted puede escribir la palabra posición brillante o normal.

La próxima instrucción es FLASH y se parece a BRIGHT en muchos aspectos. Se obtiene igual que las otras pero pulsando la tecla V. De nuevo, FLASH puede ir seguida de un 0 o un 1 para indicar si se desea el parpadeo o no. Escriba esto para ver cómo trabaja:

PRINT FLASH 1; PAPER 5; INK 1; "FLASH"

Una vez pulsado ENTER, la palabra FLASH aparecerá en la pantalla y empezará a parpadear. Como puede observar, la palabra se escribe alternativamente en azul oscuro sobre claro, y luego en azul claro sobre el fondo en azul oscuro y así continuamente. Por lo tanto, se puede ver que la instrucción FLASH, provoca que se cambien los colores del papel y la tinta una y otra vez. Es efectivo, ¿no es verdad?

La última de estas tres instrucciones referidas al color es INVERSE. La instrucción INVERSE provoca que se intercambien los colores del papel y de la tinta, es decir, hace sólo una vez lo que FLASH hace continuamente. Para obtener la palabra INVERSE se actúa como en las demás, colocando el cursor en modo E y pulsando SHIFT y M simultáneamente. Aquí tenemos a INVERSE en acción:

```
PAPER 6                  (Pulse ENTER)
CLS                      (Pulse ENTER)
PRINT INVERSE 1; INK 2; "Hola" (ENTER)
```

Después de pulsar el último ENTER se imprimirá "Hola" en amarillo sobre fondo rojo.

OIGAMOS AL SPECTRUM

¿Qué hay sobre sonido en su Spectrum? El sonido es fácil de conseguir utilizando el comando BEEP, que se obtiene de la misma manera que las últimas instrucciones y se encuentra en la tecla Z. BEEP va seguido de dos números separados por una coma. El primer número indica la duración de la nota en segundos y el segundo indica la nota. Pruebe con esto:

BEEP 1, 12

Con ello se obtiene un sonido de un segundo de duración. Para aquellos que saben de música diré que la nota representada por el número 12 es el DO de una octava superior al DO central. El número que indica la duración puede ser entero o decimal y puede variar entre 0.00125 y 10. Un sonido de duración más corto que 0,00125 segundos ya no se oye, o en todo caso parece como un zumbido. Diez segundos es el tiempo más largo que puede sonar una nota, pero las más altas sólo duran de cinco a seis segundos. Los tonos pueden ir desde —60 a + 69 (el signo "—" es necesario ponerlo pero el "+" no). El número 0 representa al DO central, por lo tanto, el 1 es el DO sostenido, el 2 es el RE, etc. Alrededor de 50 ya hay que restringir la duración a 9 segundos y cuando la nota vale 69 sólo se puede hacer sonar durante cinco segundos aproximadamente. Las notas más bajas parecen más un zumbido que música mientras que las más altas se asemejan a un trino. Así que para hacer música yo recomiendo que se utilicen sólo las dos octavas alrededor del DO central. Si usted sabe un poco de música recuerde que hay 12 semitonos en una octava, lo cual quiere decir que cada 12 números en la escala de tonos del Spectrum forman una octava. Así, el 0 es el DO central, el 12 es el próximo DO, etc.

También se admiten los tonos fraccionarios, así por ejemplo, BEEP 1,2.45 es perfectamente aceptable. Esto significa que usted puede afinar su Spectrum con cualquier otro instrumento o bien tocar música en escalas orientales.

que disponen de más de 12 semitonos en una octava, pero volveremos a esto más tarde, de momento aquí tiene una canción:

MR DOWLAND'S MIDNIGHT

```
10 BEEP 0.75,3: BEEP 0.25,4:
BEEP 0.5,5: BEEP 0.5,3:
BEEP 0.5,5: BEEP 0.5,7:
BEEP 1,4: BEEP 0.75,3:
BEEP 0.25,4: BEEP 0.5,5:
BEEP 0.5,4: BEEP 0.5,5:
BEEP 0.5,7: BEEP 1,9
```

CAPÍTULO TERCERO

Programación en BASIC con el ZX Spectrum

¿QUÉ ES UN PROGRAMA?

Anteriormente ya he mencionado en alguna parte la palabra programa, y no dudo de que usted ya la habrá oído en alguna otra parte. ¿Pero, qué es exactamente un programa? Pues bien, un programa es una serie de instrucciones usando palabras del BASIC como las que hemos ido viendo hasta ahora. La diferencia importante es que cada instrucción tiene un **número de líneas,** como esto:

```
10 PRINT "Mi nombre es Spectrum"
20 PRINT "¿Cuál es el suyo"
```

Escriba ambas líneas y pulse ENTER después de cada una. Inmediatamente observará que al contrario de lo que ocurría antes cuando usaba PRINT, el mensaje no aparece en la pantalla en el momento en que usted pulsa ENTER, sino que la línea sube arriba con una flecha ('>') delante. El interrogante se obtiene usando SYMBOL SHIFT y pulsando la tecla C.

Por tanto, la diferencia más grande que existe entre los **comandos directos** que utilizábamos hasta ahora, y las líneas de programa, es que estas últimas son almacenadas en la memoria del ordenador para ejecutarse en cuanto reciban nuestra orden. Esta orden es **RUN** que se obtiene pulsando la tecla R seguida de ENTER. Hágalo, y en la pantalla obtendrá:

```
Mi nombre es Spectrum
¿Cuál es el suyo?
```

Así, ¿qué función tienen los números de línea? Esencialmente le dicen al Spectrum que cualquier instrucción que vaya precedida de ellos debe ser almacenada y no ejecutada inmediatamente, y en segundo lugar, también sirven para indicar el orden en que deben ser ejecutadas las instrucciones. Por lo tanto, cuando usted hace RUN y ENTER el Spectrum busca el programa y empieza a ejecutar la instrucción con el número de línea más bajo. Una vez hecho esto busca el número siguiente, ejecuta la línea que le corresponde y así sucesivamente hasta que se llega al final del programa o bien se le indica que salte a un número de línea anterior, pero sobre esto ya hablaremos más adelante.

En el ejemplo se han numerado las líneas con los números 10 y 20, pero hubieran podido tener cualquier otro par de números entre 1 y 9999 mientras que el segundo sea más alto que el primero. Por ejemplo, se podrían haber usado los números 1 y 2. Pero por regla general es bueno numerar las líneas en intervalos de 10. De esta manera se podrá insertar cualquier línea que se haya olvidado sin necesidad de hacer correr todos los números de línea. Quizá la única excepción a esta regla es cuando hay que hacer un programa muy largo en el Spectrum de 48K. En este caso es posible que a intervalos de 10 se llegue a superar el máximo de 9999 antes de que se haya completado el programa. Es difícil estimar la longitud que va a tener un programa, pero intente hacerlo para determinar el intervalo más adecuado. El numerar de 5 en 5 es una buena alternativa para los programas más largos.

En mi ejemplo anterior parece que el Spectrum esté esperando una respuesta. Intente responder y vea lo que ocurre. Simplemente, intente escribir su nombre. De buen principio, ya va a tener dificultades pues la primera tecla que pulse producirá una palabra entera y no la letra correspondiente. Pero si lo consigue, al pulsar ENTER aparecerá el mensaje de error: "2 Variable not found, 0:1" que se produce cuando el Spectrum intenta interpretar su nombre o lo que haya escrito como un número. El problema está en que no hemos incluido nada en el programa que le permita al Spectrum aceptar una respuesta, y él no hará nada que no le hayamos dicho previamente mediante una línea de programa. Vamos a hacer que pueda aceptarnos una respuesta a nuestra pregunta. Esto se consigue con la

instrucción INPUT que se obtiene pulsando la tecla "I" cuando el cursor está en modo K. Escriba lo siguiente:

```
10 PRINT "Mi nombre es Spectrum'
20 PRINT "¿Cuál es el suyo"
30 INPUT n$
```

El símbolo del dólar se obtiene pulsando SYMBOL SHIFT y la tecla 4. Haga RUN y observe la diferencia. Esta vez, en lugar de dar el mensaje de que la tarea ha terminado, el Spectrum presenta el cursor L parpadeante entre dos comillas en la parte inferior izquierda de la pantalla. Escriba su nombre y pulse ENTER. Su nombre apareció entre las comillas pero tan pronto como pulsó ENTER desapareció y fue reemplazado por el mensaje "0 OK" otra vez. Ahora observe, escriba esto tal como está aquí:

PRINT n$ (y pulse ENTER)

Su nombre aparecerá justo debajo de la última cosa que hay escrita en la pantalla (que probablemente era la frase "¿Cuál es el suyo"). ¿Ha visto lo que ha ocurrido?, el Spectrum ha asignado su nombre a n$ en la instrucción PRINT y cuando le hemos dicho que escribiera su nombre lo ha hecho obedientemente. A n$ se le llama **variable de cadena** y puede ser cualquier letra del alfabeto seguida del símbolo del dólar. El símbolo del dólar detrás de una letra le indica al Spectrum que con ese nombre (por ejemplo n$) reconocerá de ahora en adelante una **cadena** de caracteres. En lugar de su nombre usted podía haber escrito cualquier otra cosa, inténtelo, pero antes añadiremos una línea más al programa:

```
40 PRINT "Hola"; n$      (indica un espacio)
```

Cuando haga RUN el Spectrum le dirá hola a usted, o a cualquier cosa que ponga en el momento de preguntarle su nombre. Experimente con distintos nombres, por ejemplo, "123 mmsgdter4tre" o cualquier otra cosa así de extraña y verá que el Spectrum no distingue, todo lo que le escribe lo enseña luego como si fuera su nombre. Vamos a ver rápidamente qué es lo que hace este programa y cómo lo hace. Primero pulse ENTER para obtener el listado del

programa, que sustituirá a los mensajes que había en la pantalla.

Las primeras dos líneas simplemente escriben en la pantalla dos mensajes. La línea 30 hace que el Spectrum se espere hasta que usted entre una cadena de caracteres (que debería ser su nombre, pero ya hemos visto que acepta cualquier cosa). Todo lo que usted escriba en la respuesta hasta que pulsa enter es tomado por el Spectrum y etiquetado con el nombre n$. En la línea 40 el Spectrum escribe "Hola" e inmediatamente después, escribe la cadena que ha sido etiquetada con el nombre n$, y no los caracteres n y $ (para hacer esto último, deberían estar entre comillas ¿recuerda?). Ahora vamos a añadir algunas líneas al programa que lo harán más divertido, o quizá más embarazoso, depende de la edad que tenga usted.

```
40 PRINT "Yo soy joven".                (Escriba las
   "¿Cuántos años tiene usted?"         dos comillas.)
50 INPUT edad
60 PRINT n$; "tiene"; edad; "años"
```

Haga RUN y observe los cambios, esta vez, el Spectrum enseña cosas como éstas:

```
Mi nombre es Spectrum
¿Cuál es el suyo?
Yo soy muy joven
¿Cuántos años tiene usted?
Tim tiene 189 años
```

Y ahora, para hacerlo más interesante añada estas dos líneas:

```
55 CLS       (se encuentra en la tecla V)
70 GOTO 60   (pulse G para obtener GOTO
```

Intente adivinar qué es lo que ocurrirá, para ello recuerde que CLS borra la pantalla y que **GOTO** significa "IR A", es decir, produce un salto a la línea que se le indica. Haga RUN del nuevo programa.

¿Sorprendido? Ha aparecido un "¿scroll" en la parte de abajo de la pantalla. Si quiere continuar pulse cualquier tecla que no sea SPACE o N. Ha observado también que

le he hecho escribir la línea 40 dos veces. Pulse ENTER y mire cuál de las dos es la que está. Sólo está la última que se escribió. Recuerde esto, pues es muy útil saber que al escribir una misma línea dos veces se borra la que había anteriormente. Si usted comete un error y la línea aún está abajo entonces usted puede usar DELETE (CAPS SHIFT y 0) para corregirla. Si ya ha pulsado ENTER entonces puede escribirla de nuevo para hacer la corrección.

¿Cómo trabaja el programa? En la línea 40 el Spectrum pregunta por su edad y en la línea 50 se espera a que la introduzca y la carga en una *variable* numérica que yo he querido que se llamara "edad", veremos que las variables numéricas (que son aquéllas que sirven para almacenar números en lugar de caracteres en general) no llevan detrás el signo del dólar. Es posible utilizar una palabra entera como nombre de *variable numérica,* mientras que las de cadena sólo pueden tenerlo de una sola letra del alfabeto (seguida de $). Comprenderá que he elegido n$ para el nombre para recordar que era para el Nombre. Es muy provechoso que coja el hábito de nombrar las variables con nombres que recuerden a su contenido. El hecho de usar "edad" para la edad está mucho mejor que haber usado "e", y esto último es mejor que usar "x", que no tiene nada que ver con el contenido y hará que no se comprenda el programa si se repasa algunos meses más tarde.

Por último, la línea 60 escribe el nombre que el Spectrum tiene almacenado en n$, inmediatamente escribe "tiene" con un espacio en cada lado, luego escribe el contenido de la variable edad y seguidamente la palabra años con un espacio en cada lado. ¿Sabe por qué he puesto un espacio después de "años"? ¿y un punto y coma? Pues bien, el punto y coma hace que la próxima vez que se escriba algo lo haga a continuación de lo anterior, y el espacio es para que quede un hueco entre los dos campos que se imprimen. La línea 70, como no dudo que habrá adivinado, envía al Spectrum de vuelta a la línea 60 una y otra vez para escribir las mismas series de palabras y números.

Vamos ha intentar un programa más difícil. En él se pregunta por un año del pasado y el Spectrum le da una idea de cuántos años, meses, horas y minutos han pasado hasta el año actual. Escriba el programa y haga RUN:

```
   10 REM "CUANTO TIEMPO?"
   20 INPUT "Anno en el pasado";
y1
   30 INPUT "Anno actual";y2
   40 LET yrs=y2-y1
   50 LET mth=yrs*12
   60 LET dys=yrs*365
   70 LET hrs=dys*24
   80 LET mins=hrs*60
   90 PRINT "Han pasado ";yrs;" annos,
"'';mth;" meses,"'';dys;" dias,"'';
hrs;" horas,"''"y ";mins;" minutos"
```

La línea 90 es la más difícil, especialmente hasta que usted no domine todos los pequeños trucos que encierra. En todo caso, intente por favor reescribirla exactamente tal y como está aquí, con cada espacio, cada comilla, cada punto y coma y cada par de apóstrofes. Todo es necesario (el apóstrofe se obtiene con SYMBOL SHIFT y 7). Ahora, si usted ha hecho RUN del programa obtendrá una salida en la pantalla más o menos como esto:

Anno en el pasado 1604	(parte baja de la pantalla)
Anno Actual 1982	(parte baja de la pantalla)
Han pasado 378 annos,	(arriba)
4536 meses,	
137805 días,	
3311280 horas	
y 1.986768E + 8 minutos.	(Esto puede parecerle un poco extraño, pero ya explicaré la manera que tiene el Spectrum de escribir los números más grandes.)

En este programa he introducido muchas cosas nuevas, intencionadamente, así que no se preocupe si lo encuentra un poco difícil. Vamos a verlo línea por línea.

De entrada tenemos una nueva palabra en la línea 10. **REM** es una palabra del BASIC que es una abreviación de la palabra inglesa REMARK que significa comentario. Esta línea no hace nada más que dar una aclaración sobre lo que hace, o de qué trata el programa o una parte del programa. En una sentencia REM se puede poner tanto un título, como una breve descripción, un aviso, etc. Es fácil escribir una sentencia REM. La palabra REM se obtiene pulsando la tecla E y ya se tiene en la pantalla. El Spectrum no hace ningún caso de esta línea cuando se ejecuta el programa. Para probarlo escriba:

10 y pulse ENTER

Esto provoca que la línea 10 sea borrada del programa. Haga RUN para provar que no hay ninguna diferencia. Es una buena práctica usar sentencias REM en los programas que ayuden a recordar qué es lo que hace, cómo lo hace, etc. ¡Es como un salvavidas encontrarse sentencias REM en un programa largo y complicado cuando se vuelve a él después de un tiempo!

¿Qué hay de las líneas 20 y 30? Parecen un cruce entre las líneas con INPUT que vimos antes y las sentencias con PRINT. De hecho esto es una buena descripción de lo que son en realidad. Cuando usted quiere que su Spectrum le pregunte algo por pantalla, tiene que usar previamente una sentencia PRINT para que aparezca la pregunta. Pero para obtener la información hay que usar una sentencia INPUT en la que el Spectrum se espera a que escriba un número o una palabra, que es asignado a una variable numérica o de cadena. Gracias a este nuevo tipo de sentencia INPUT podemos poner juntas en una misma línea las dos cosas, la aparición de la pregunta, y el tratamiento de la respuesta. Este método es más elegante y ocupa menos espacio, pero ésta no es la única diferencia. Para ilustrarlo, escriba una nueva línea 20 y añada una línea 21, dejando la línea 30 tal como está para poder comparar los dos métodos. Escriba:

```
20 PRINT "Anno en el pasado?"
21 INPUT Y1
```

Ahora haga RUN del programa otra vez para ver los cambios. En primer lugar la pregunta "año en el pasado" aparece arriba de la pantalla en vez de abajo, pero en cambio la respuesta se escribe igualmente abajo. Es cuestión de opinión decidirse por uno de los dos métodos en un momento determinado, pero el hecho de disponer de los dos da más flexibilidad a la programación. El método de INPUT "..." deja la pantalla limpia mientras que el otro mantiene la pregunta en la pantalla (aunque también puede colocar un CLS en la línea 80 más o menos, para deshacerse de ella). También puede escoger entre poner una coma o un punto y coma después del mensaje o pregunta de una línea INPUT. Debe ir después de las últimas comillas y antes de la variable. Pruebe esto como comandos directos:

```
INPUT "Cuánto es", q$        (y pulse ENTER)
INPUT "Cuánto es", q$        (y pulse ENTER)
```

Como puede ver, la única diferencia, es que el cursor L (y por lo tanto, el lugar donde se escribirá su respuesta) está en el primer caso en el lado derecho de la parte baja de la pantalla mientras que en el segundo caso, está justo detrás de la pregunta. Usted puede usarlo como quiera, aquí sí que no hay reglas.

Ahora usted ya debe haber comprendido algo de lo que hacen la coma y el punto y coma en las sentencias PRINT e INPUT. El punto y coma hace que lo próximo que se vaya a imprimir lo haga justo detrás de lo último que se imprimió. Por otro lado la coma indica que el próximo campo a imprimir debe hacerlo en la mitad derecha de la pantalla, pero si ésta ya tiene alguna cosa escrita entonces lo hace en la siguiente línea. El punto y coma coloca la información seguida mientras que la coma va muy bien para poner texto o números en dos columnas, derecha e izquierda. Por ejemplo:

```
PRINT "Hola", "Aquí"     (y ENTER)
PRINT "Hola"; "Aquí"     (y ENTER)
```

Volvamos al programa. Las líneas de la 40 a la 80 calculan el número de años, meses, días, horas y minutos que han pasado desde el año que usted da en la línea 20 hasta el que introduce en la línea 30. He dado a cada variable un nombre que es una abreviación de lo que contienen. Como puede ver el programa simplemente calcula el número de meses multiplicando por 12 el número de años, los días multiplicando por 365 el número de años, y así sucesivamente.

La línea 90 es la más difícil de introducir y requiere plena concentración para ser entendida completamente. En esencia, lo que hace es escribir en la pantalla el número de años, meses, días, etc., que se han calculado. Pero como he pensado que será mucho más fácil leer la respuesta, si ésta está en líneas separadas, ha habido que incluir todos estos signos de puntuación: comas, puntos y comas, apóstrofes. El apóstrofe hace que el próximo campo a imprimir lo haga en la próxima línea, por lo tanto, dos apóstrofes saltarán dos líneas. Ahora observe con atención la línea 90 y podrá ver fácilmente por qué cada respuesta se escribe en una línea separada.

Problema: Usando la información que se ha dado en esta última parte, usted debe ser capaz de mejorar un poco el programa. Por ejemplo, el programa no permite tener en cuenta los días, ni los meses de los dos años. ¿Puede añadir algunas líneas para arreglarlo? El programa también ignora los años bisiestos, pues haga algo para que se tengan en cuenta en los cálculos. También puede hacer que calcule el número de segundos que han pasado.

Finalmente, estará un poco sorprendido de haber obtenido los minutos que han pasado en una forma como ésta:

1,986768E + 8

¿Qué significa esto? Pues bien, si usted ha hecho matemáticas, habrá reconocido la **notación científica.** Esta manera de escribir los números divide la información en dos partes, la primera es siempre un número decimal con un solo dígito antes de la coma; la segunda parte, indica la potencia de 10 por la que hay que multiplicar a la primera para obtener el número en cuestión. Por si no ha quedado claro, lo veremos con un ejemplo. Tomemos el número 1000. En notación científica se escribe:

1,0E + 3

Esto significa que hay que multiplicar 1,0 por 10 elevado a tres. La E mayúscula le dice al Spectrum que lo que le sigue es un exponente. El signo más indica que la potencia es positiva. Como probablemente ya sabrá usted, es lo mismo decir que elevamos 10 una potencia X que poner X ceros detrás de un 1. Así 10 elevado a 3 es 10^3 que es 1000. Por otro lado, 10^{-3} es 0,001. ¿Cuántos minutos hemos obtenido antes?, pues bien, hay que multiplicar el número 1,986768 (casi es 2) por un 1 seguido de ocho ceros (ya que tenemos E + 8), es decir, 1,986768 × 100.000.000.

¡GUARDÉMOSLO!

Ahora tiene un programa en la memoria del Spectrum. ¿Cómo guardarlo en un cassette? Después de todo supongo que usted no querrá escribir el programa de nuevo cada vez que vaya a utilizarlo. Es muy simple guardar un programa en el cassette. Primero asegúrese de que dispone del cable con dos enchufes (en cada lado), uno gris y el otro negro. Enchufe el gris (podía haber sido el otro pero he escogido el gris) en la entrada MIC (para grabar) de su cassette: Ahora enchufe el otro gris en la salida marcada MIC que está en la parte de atrás del Spectrum. Asegúrese de que dispone de una cinta virgen y córrala hacia atrás hasta el principio. Ahora córrala un poco hacia adelante hasta que la parte marrón de la cinta esté situada delante de los cabezales (esto puede verlo sacando la cinta). Su cassette probablemente dispone de nivel automático de grabación, pero en caso contrario sitúe el volumen a tres cuartos aproximadamente del máximo. Ahora todo está listo para efectuar la grabación del programa. Con la cinta ya preparada pulse la tecla S y la palabra SAVE aparecerá en la pantalla. Después de SAVE el Spectrum espera el nombre de su programa encerrado entre comillas. El comando SAVE debe parecerse a esto:

SAVE "Nombre"

Cuando pulse ENTER el Spectrum escribirá un mensaje en la pantalla que le indica que debe poner su cassette en

marcha para grabar. Seguidamente aparece el mensaje "press any key" que significa que pulse cualquier tecla. Hágalo y en el borde de la pantalla aparecerán líneas rojas y azules corriendo hacia arriba. Esto ocurre dos veces, primero cuando se graba el nombre del programa y luego cuando se graba el programa mismo, pero ahora con líneas amarillas, azules y negras. En ese momento el Spectrum también hace un zumbido. Cuando se termina la grabación aparece el mensaje "0 OK". El programa está ahora guardado en la cinta y ya puede parar el cassette.

Su programa se habrá grabado bien (al contrario que el ZX81, el Spectrum es muy fiable a la hora de grabar programas). Pero además usted puede comprobarlo usando el comando **VERIFY.** Rebobine la cinta hasta el punto en que se encuentra el principio del programa que acaba de grabar. Ahora enchufe dos de las clavijas del mismo color, una detrás del Spectrum en la entrada marcada EAR y la otra en el cassette en la salida marcada EAR o MONITOR, en caso de que no disponga pruebe con la salida para auriculares o altavoz externo. Ahora escriba VERIFY, pulsando CAPS SHIFT y SYMBOL SHIFT al mismo tiempo y luego la tecla R. Detrás escriba el nombre de su programa entre comillas. Este nombre debe ser exactamente el que usted le dio al programa cuando lo grabó, incluso los espacios, las minúsculas y las mayúsculas. Tampoco puede contener más de 10 caracteres. Ahora pulse ENTER y pulse la tecla Play de su cassette. El Spectrum irá mostrando el nombre de todos los programas que encuentre en la cinta. Cuando llegue al programa que se desea verificar mostrará de nuevo aquellas rayas amarillas y azules que aparecieron cuando lo grabó. Cuando haya terminado aparecerá un mensaje que normalmente es el ya conocido "0 OK". Si algo anda mal, probablemente dirá "Tape loading error", en caso de que esto ocurra compruebe que el nombre que ha dado sea el correcto así como el nivel de volumen. Haga la misma operación en caso de que no esté seguro, y si hay problemas, de nuevo, grabe el programa otra vez.

¿Cómo recuperar el programa desde la cinta al ordenador? Bien, esto también es fácil. Esta vez debe usar el comando **LOAD.** Primero rebobine la cinta hasta el principio del programa (sin dunda es mejor que vaya más hacia atrás). Una vez esté la cinta colocada, enchufe las clavijas

en las hembras marcadas EAR (lo mismo que en VERIFY). Ahora pulse la tecla J y la palabra LOAD aparecerá, ponga detrás el nombre exacto del programa entre comillas. Pulse ENTER y ponga en marcha la cassette como si fuera a escuchar lo que hay en ella. El borde de la pantalla se llenará de líneas rojas y azules hasta que el programa cuyo nombre ha dado sea localizado. Cuando esto ocurre el color de las líneas cambia a amarillo y azul. Al terminar aparece el informe "0 OK" y ya puede apagar el cassette. Observe que las líneas que se escriben para verificar (VERIFY) y para recuperar (LOAD) son muy similares:

VERIFY "programa'
LOAD "programa"

Recuerde que su Spectrum es muy exigente: Si ha usado letras mayúsculas cuando hizo SAVE debe también usarlas en LOAD y VERIFY. Lo mismo ocurre con las minúsculas.

SI NO PUEDE RECORDAR EL NOMBRE DEL PROGRAMA

Entonces use esta expresión para recuperarlo:

LOAD ""

No deje espacios entre las comillas. Al pulsar ENTER entonces el Spectrum recuperará el primer programa completo que encuentre. Si no es el que desea entonces pulse NEW (aunque no es necesario) y vuelva a intentar lo mismo con el siguiente. Por supuesto que si no puede recordar el nombre de su programa ni el lugar de la cinta en que éste se encuentra es bastante engorroso ir repitiendo LOAD "" cada vez. Aquí tiene un truco para evitarlo:

Para obtener un catálogo de lo que hay en una cinta, rebobine ésta hasta el principio y escriba lo siguiente:

VERIFY "CAT" (y ponga en marcha la cassette igual que en LOAD)

Usted puede usar cualquier otra palabra que no sea CAT pero que no coincida con el nombre de ningún programa de la cinta. El Spectrum recorrerá la cinta entera mostrando los nombres de los programas, pero sin recuperar ni verificar ninguno de ellos. Una razón por la que he utilizado la palabra CAT para este propósito es que otros ordenadores disponen de un comando CAT que hace exactamente lo mismo.

MERGE

Probablemente ha visto esta palabra en el teclado y se ha preguntado para qué sirve. Pues bien, con **MERGE** usted puede recuperar de la cinta dos o más programas diferentes al mismo tiempo en memoria. Si todos tienen distintos números de línea y juntos no exceden del total de memoria disponible, entonces todos los programas recuperados con MERGE podrán "convivir" juntos en la memoria del Spectrum. Por ejemplo, supongamos que escribimos estos tres programas:

```
1 REM PROGRAM
10 PRINT "Hi"

2 REM PROGRAM 2
20 PRINT "There"

3 REM PROGRAM 3
30 PRINT "Fred"
```

Si recuperamos el programa 1 y luego hacemos MERGE con los programas 2 y 3 (por supuesto habiéndolos grabado previamente en la cinta como programas separados) nos encontremos con esto en la memoria del Spectrum:

```
1 REM PROGRAM 1
2 REM PROGRAM 2
3 REM PROGRAM 3
10 PRINT"Hi"
20 PRINT"There"
30 PRINT"Fred"
```

Si por ejemplo hubiéramos usado la línea 2 en más de un programa, entonces la última línea dos que entrara en memoria provocaría que se borrara la anterior. Por lo tanto, si quiere unir dos programas procure que tengan números de línea diferentes. Un uso muy útil de MERGE es para añadir rutinas cortas al final de la memoria (digamos de la línea 9000 en adelante). Por ejemplo, más adelante en este libro habrá una rutina de renumeración que permitirá cambiar los números de líneas de los programas.

LOS BUCLES - ALGO NUEVO

Volvamos a programar. Pero usted probablemente tenga algún programa en memoria. Para deshacerse de él y empezar de nuevo use **NEW.** El uso de NEW es bastante drástico, así que es importante el vigilar no escribirlo por error. Pulsando la tecla A cuando el cursor está en modo K, aparecerá NEW en la pantalla. Si luego pulsa ENTER la pantalla ennegrecerá durante unos instantes y luego estará limpia, con el mensaje "Sinclair Research" que aparece cuando se enchufa. Por lo que al programa se refiere, el uso de NEW equivale a desenchufar y volver a enchufar, por lo que hay que tener cuidado cuando quiera utilizar la letra A o el comando STOP. Recuerde: al hacer NEW se pierde el programa.

Ahora vamos a por los bucles. Recordará que hace poco hicimos un programa que escribía varias veces su nombre y edad. Para ello usamos GOTO que puede enviar el programa a una línea anterior. Pero ¿qué pasa si queremos poner un nombre en la pantalla exactamente quince veces? O bien, ¿si queremos que se pregunten siete cuestiones y que el Spectrum almacene las respuestas? Bien, pues para hacer esto usamos las palabras **FOR** y **TO** junto con **NEXT** para crear un bucle. Aquí tenemos un ejemplo:

```
10 REM CONVERTIDOR DE LITROS
20 FOR N = 1 TO 20
30 INPUT "¿CUANTOS GALONES?", G
40 LET L = 4.546*G
50 PRINT G;"∧GALONES SON∧;L;"∧LITROS"
60 NEXT N
```

La línea 10 es una sentencia REM que nos recuerda qué es lo que hace el programa. La línea 20 es el principio del bucle en el que se usan las palabras FOR y TO para decirle al Spectrum cuántas veces queremos que se repita el proceso que está en la línea 30 a 50. Como puede ver he decidido que ocurra 20 veces, ya que en la línea del FOR... TO la variable N toma valores del 1 hasta el 20. Las líneas 30 a 50 comprenden la conversión de galones a litros. En la línea 30 se usa la ya familiar sentencia INPUT con una frase que pregunta cuál es el volumen en galones que se quiere convertir a litros. La línea 40 calcula el número de litros, para ello crea una nueva variable llamada L (por Litros, por supuesto) y le dice al Spectrum, mediante la instrucción LET, que L es igual a G (volumen en galones) multiplicado por 4,546. La línea 50 simplemente escribe el resultado en una frase. La línea 60 es la otra parte del bucle y hace que el valor de N aumente en uno hasta un máximo de 20. Esta línea dice "vuelve a la línea en que está el FOR (la 20 en este caso), cogiendo el próximo (en inglés, NEXT) valor de N a no ser que ya haya llegado al máximo (que en este caso es 20)". Cuando se llega a 20 veces, entonces se habrán convertido 20 valores en galones a su equivalente en litros y el programa se detendrá con el informe:

0 OK, 60:1

Supongo que habrá visto que es muy fácil conseguir que el Spectrum le haga la pregunta una sola vez o 100 veces, o cualquier número de veces que usted quiera. Lo único que hay que hacer es cambiar el 20, que hay al final de la línea 20, por otro número.

EDICIÓN

Vamos a cambiar este número por cinco para ver lo fácil que es hacerlo. La manera más simple es usar una nueva función llamada **EDIT**, que le permite traer una de las líneas del programa a la parte baja de la pantalla, cambiar o borrar cualquier trozo de línea y luego devolverla al programa.

Para encontrar EDIT pulse CAPS SHIFT y la tecla 1 al mismo tiempo. La última línea de programa que escribió,

bajará a la parte inferior de la pantalla. Ahora pulse ENTER y aparecerá un listado del programa sin ningún cambio. Para hacer un cambio en la línea 20, primero tiene que buscar en qué línea está el cursor'>' que estará junto a la última línea que introdujo. Si hacemos EDIT nos bajará la línea que tiene este cursor por eso debemos colocarlo antes en la línea 20. Para hacerlo se usan las flechas de dirección que las encontrará sobre las teclas 5 a 8. De hecho la que tendrá que utilizar es la flecha hacia arriba que se encuentra sobre la tecla 7. Para mover el cursor (de ahora en adelante lo designaré como *cursor de línea actual)* pulse CAPS SHIT y la tecla 7 simultáneamente. Cada pulsación provoca que el cursor de línea actual suba hacia la línea inmediatamente superior. Unas pocas pulsaciones y ya estará sobre la línea 20 (o simplemente mantenga la tecla apretada, pues también dispone de autorrepetición). Antes de bajar la línea 20 haga pruebas moviendo el cursor de línea actual arriba y abajo (con la tecla 6).

Con el cursor de línea actual al lado de la línea 20 pulse CAPS SHIFT y 1 y la línea en cuestión bajará. Ahora, para mover el cursor L sobre la línea pulse CAPS SHIFT y 8 hasta llegar justo detrás del 20. Borre el 20 con CAPS SHIFT y 0 (DELETE). Dos pulsaciones con DELETE y el 20 desaparecerá. Para poner el 5 en sustitución del 20 simplemente escríbalo y pulse ENTER. Inmediatamente aparecerá un nuevo listado en el que la línea 20 estará así:

```
20 FOR N =    TO 5
```

Haga RUN del programa para comprobar que ahora el bucle sólo se realiza 5 veces.

Aquí hay dos consejos para utilizar EDIT:

CONSEJO

Primero: Si usted tiene una línea muy larga y acaba de cambiar algo del final de la línea, pero luego se da cuenta de que hay un error en el principio entonces es más rápido pulsar ENTER y volver a hacer EDIT que no correr el cursor hacia atrás con la tecla 5. Por supuesto que esto

sólo funciona si el error que hay le permite pulsar ENTER sin que aparezca el error de sintaxis.

CONSEJO

Segundo: Habrá observado que cuando tiene un programa muy largo del cual quiere editar una línea, para colocar el cursor de línea actual en dicha línea puede usar LIST × (donde × es el número de línea). Pero esto presenta el problema de que produce un listado que llena la pantalla y entonces el Spectrum pregunta si queremos continuar ("Scroll"). Para evitar el ser preguntado coloque el cursor en la línea deseada escribiendo un número de línea menos y pulsando ENTER. Esto suponiendo que dicho número no exista pues sino se borraría (de todas maneras es improbable si usted me ha hecho caso y ha numerado sus líneas de 10 en 10). Esto produce un listado con el cursor de línea actual en la línea deseada y sin la pregunta "scroll". Por ejemplo:

```
800 PRINT A$
810 LET C = S + 2
```

Si tiene estas líneas en medio de un programa y ha visto que C debe ser igual a "S + 3" y no a "S + 2" entonces escriba:

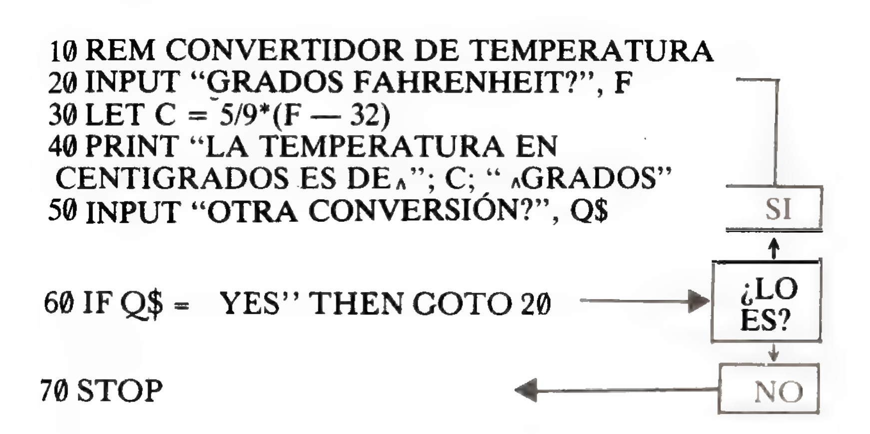
809 (y pulse ENTER)

Como que la línea 809 no existe entonces el cursor de línea actual se coloca en la siguiente que es la 810 que ya puede ser bajada rápidamente con EDIT.

SI... ENTONCES...

¿Qué ocurre si usted no está seguro de cuántas veces quiere convertir galones a litros? O de ¿cuántas veces quiere repetir cualquier proceso? Pues bien, es muy fácil resolver esto utilizando las **sentencias condicionales.** Para ello

se usan las instrucciones **IF** (que en español significa "si" condicional) y **THEN** (que significa "entonces"). Estas dos palabras clave junto con GOTO resuelven el problema fácilmente. Observe esta versión del programa de conversión, que esta vez convierte grados Fahrenheit a grados Centígrados.



En el lado derecho he puesto un pequeño esquema para que vea qué es lo que el Spectrum decide en cada momento. Del mismo modo que el programa anterior, éste pregunta por la temperatura en grandos Fahrenheit, y luego, en la línea 30 calcula su equivalente en grados Centrígrados. La respuesta se imprime en la lína 40. Observe que he vuelto a utilizar "∧" para indicar los espacios.

La línea 60 es nuestra sentencia condicional y actúa según la respuesta que demos en la línea 50 en que se pregunta si queremos hacer otra conversión. En respuesta usted puede contestar YES o NO. Esta respuesta es asignada a la variable Q$ (recuerde que el signo "$" le indica al Spectrum que la variable es de cadena, es decir que contiene palabras o letras). En la línea 60 se pregunta si Q$ es YES o no lo es, y en caso de que lo sea vuelve a empezar (THEN GOTO 20).

Si usted no responde YES, entonces continúa y se encuentra en la línea 70 con la instrucción **STOP**, que como ya puede suponer produce la detención del programa. De hecho el programa se hubiera detenido de todas maneras, ya que no había más líneas. Al usar STOP se produce el siguiente mensaje:

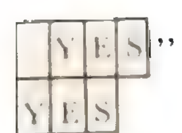
9 STOP statement, 70:1

En algunos ordenadores es obligatorio terminar un programa con STOP o con END pero el Spectrum no lo requiere. De todos modos, STOP puede ser muy útil en algunas ocasiones, pues no sólo provoca la detención del programa sino que también nos dice dónde tuvo lugar (en este caso ha sido en la línea 70, primera parte).

Recordará que anteriormente ya he dicho que los ordenadores pueden ser muy exigentes. Al comparar cadenas tal como hemos hecho en el programa es uno de los puntos en que la exigencia del Spectrum puede llegar a confundirle. Por ejemplo, si en vez de escribir "YES" hubiera escrito "^YES", es decir, con un espacio antes de la palabra, usted o yo hubiéramos dicho que se trataba de la palabra YES mientras que el Spectrum que es muy estricto diría que no lo es. Para comparar la respuesta con las letras YES el Spectrum coloca una al lado de otra para ver si son exactamente lo mismo:

Su respuesta:

Comparada con:



Ya puede ver que mientras cualquiera de nosotros diríamos que las dos palabras son lo mismo, para el Spectrum no se parecen en absoluto ambas cadenas de caracteres. Lo primero que observa es que la respuesta tiene longitud cuatro mientras que YES tiene longitud tres. Y además, donde debería encontrar una Y se encuentra con que hay un blanco, donde debería haber una E encuentra una Y, etc. En definitiva intente recordar siempre lo exigentes que son los ordenadores o perderá la cabeza enfrentándose con problemas de fácil solución.

¿Qué pasa si usted, no contesta ni YES ni NO? Pues bien, la sentencia condicional de la línea 60 sólo comprueba si la respuesta es YES o no es YES, por lo tanto, cualquier otra respuesta, incluso ninguna respuesta hará que el Spectrum actúe como si la respuesta fuera NO. Piense cómo lo haría para añadir una línea, la 65 por ejemplo, que imponga la condición de que si la respuesta no es ni YES ni NO vuelva a efectuarse la pregunta.

Hemos visto cómo hacer qûe un bucle realice un proceso un cierto número de veces y también cómo hacerlo de manera que finalice en el momento en que queramos. Pero supongamos, por ejemplo, que queremos escribir un símbolo en la pantalla un cierto número de veces pero cada símbolo separado por una distancia. Este problema puede ser resuelto bajo algunas circunstancias usando la instrucción **STEP** en los bucles FOR NEXT. Aquí tenemos a STEP en acción:

```
10 BORDER 4
20 INK 2
30 PAPER 6
40 CLS
50 FOR N = 1 TO 31 STEP 2
60 PRINT TAB N; "↑";
70 NEXT N
```

Haciendo RUN de este programa se obtiene una línea de flechas en la parte de arriba de la pantalla, separadas cada una por un espacio. El secreto de todo esto está en el uso de la función **TAB.** TAB x, hace qué lo que se imprima lo haga en la columna x (x espacios a la derecha) de la línea en que se imprime. El valor máximo de x es 31, pues las columnas se numeran desde 0 a 31. Si usted da un número mayor de 31 entonces el Spectrum le resta 32, de manera que siempre se imprime en la línea en cuestión. Así, TAB 32, no imprimirá en la primera posición de la línea siguiente sino en la primera de la línea presente, a condición de que no haya nada escrito en dicha línea.

```
PRINT TAB 32 = PRINT TAB 0
PRINT TAB 33 = PRINT TAB 1
```

Continuemos, cambie la línea 50 por:

```
50 FOR N = 1 TO 31 STEP 4
```

Haga RUN del programa y compruebe lo que ocurre al cambiar el paso (STEP en inglés).

Para colocar algo en una posición determinada de la

pantalla también puede usar **PRINT AT**, que tiene la ventaja de que puede imprimir en cualquier columna de cualquier fila, y no sólo en cualquier columna de la fila en que se está imprimiendo como TAB. Aquí tiene como una sentencia PRINT AT reemplaza a la sentencia TAB del programa anterior:

```
60 PRINT AT 0,N;"↑"
```

PRINT AT, siempre va seguido de dos números separados por una coma. El primero especifica la fila (de 0 a 22) y el segundo, la columna (de 0 a 31). Como puede ver, aunque en el programa se ha especificado la línea 0, hubiera sido muy fácil, poner cualquier otra línea cambiando el primer número detrás de PRINT AT. Intente hacer un bucle FOR NEXT que imprima una línea de flechas en cualquier otra fila de la pantalla.

¿Puede imaginarse el efecto que causaría el borrar el punto y coma de la línea 60? Bórrelo usando EDIT, para ver lo que pasa.

Para los incansables, aquí hay otra versión del programa que usa TAB otra vez. Mire si puede ver lo que hace.

```
 5 LET X = 1
10 FOR N = 1 TO 31 STEP X
20 POKE 23692, - 1 (esto provoca un scroll automático)
30 PRINT TAB N;"↑"
40 NEXT N
50 LET X = X + 0.5
60 GOTO 10
```

Se encontrará con que tiene que parar el programa con **BREAK,** ya que la línea 60 provoca un bucle continuo hacia la línea 10. Para hacerlo pulse CAPS SHIFT y la tecla SPACE/BREAK simultáneamente.

LÍNEAS CON VARIAS INSTRUCCIONES

En el programa de la línea de flechas, empezamos cambiando la tinta (INK), el papel (PAPER), y el borde (BORDER). En total, este proceso, junto con el obligado CLS nos ocupaba cuatro líneas de programa. Es posible

hacerlo todo en una línea. El Spectrum permite varias instrucciones en una sola línea separadas por dos puntos ":". En efecto, cada instrucción es como una línea de programa, pero distintas instrucciones tienen el mismo número de línea. Observe cómo se coloca en una línea las asignaciones de color del programa que hablábamos:

```
10 INK 2:PAPER 6: BORDER 4: CLS
```

Recuerde que este tipo de sentencias no es *exactamente* lo mismo que tener varias líneas en una. No se puede por ejemplo ir (GOTO) a una instrucción de una línea múltiple sin ejecutar las otras.

Hay que vigilar que tipos de instrucciones se mezclan en una línea. No todas las combinaciones funcionarán satisfactoriamente. Por ejemplo, es una buena idea colocar sentencias REM que indiquen lo que hace una línea o una parte del programa, pero nunca coloque una sentencia REM al principio de una línea múltiple, pues el resto de la línea no se ejecutará nunca ya que será tomado como parte del comentario. Compare por ejemplo estas dos líneas:

```
10 REM no lo haga así: PRINT "MAL"
20 PRINT "BIEN": REM así se debe hacer
```

Si hace RUN de este programa, sólo se imprimirá "BIEN". Las sentencias condicionales, también pueden causar problemas. No coloque sentencias IF... THEN al principio o en medio de líneas múltiples, ya que el Spectrum ignorará las instrucciones que le siguen si la condición no se cumple. Compare éstas:

```
10 IF 2>3 THEN PRINT "YES" :PRINT "hi"
20 PRINT "hello": IF 2<3 THEN GOTO 40
30 PRINT " at 30":STOP
40 PRINT "at 40" :STOP
```

Como verá al ejecutar el programa, sólo se imprime la palabra "hello" y el programa se para con el STOP de la línea 40; demostrando que si la condición se cumple, entonces se ejecutan las demás instrucciones de la línea. Tenga en cuenta que el uso de las líneas múltiples dificultará el

seguimiento del programa para otras personas o incluso para usted al cabo de un tiempo, así que úselas con moderación y claridad.

IR Y VOLVER

Hemos usado GOTO en varias ocasiones, pero tiene un hermano llamado **GOSUB,** que hace una función similar y puede ser muy útil en ocasiones. Primero veamos a GOTO de nuevo en acción. Escriba el siguiente programa y ejecútelo:

```
 10 GOTO 100
 20 GOTO 200
 30 GOTO 300
 40 STOP
100 PRINT "EN LINEA 100"
110 GOTO 20
200 PRINT "EN LINEA 200"
210 GOTO 30
300 PRINT "EN LINEA 300"
310 GOTO 40
```

Al hacer RUN deberá obtener:

```
EN LINEA 100
EN LINEA 200
EN LINEA 300
```

Ahora crearemos una rutina muy similar usando GOSUB y RETURN en lugar de GOTO:

```
 10 GOSUB 100
 20 GOSUB 200
 30 GOSUB 300
 40 STOP
100 PRINT "EN 100": RETURN
200 PRINT "EN 200": RETURN
300 PRINT "EN 300": RETURN
```

La diferencia más importante es que GOSUB salta a un número de línea y continúa el programa hasta que encuen-

tra un RETURN (en español "vuelta atrás"). En cuanto el RETURN es localizado, el programa vuelve a la línea siguiente a GOSUB. Dado que RETURN no incluye ninguna dirección de vuelta, usted puede intercambiar las líneas 10 y 20 y el programa trabajará igualmente. Use EDIT para efectuar el cambio y cuando ejecute la nueva versión obtendrá:

```
EN 200
EN 100
EN 300
```

Con esto se demuestra que efectivamente el GOSUB hace que el Spectrum guarde la dirección de retorno. Aquí tenemos un programa que usa GOSUB de un modo más práctico:

```
 10 BORDER 2: INK5: PAPER 1: CLS
 20 INPUT "¿Cuanto son 3 × 6?", ×
 30 IF × 18 THEN GOSUB 100
 40 IF × <> 18 THEN GOSUB 200
 50 INPUT "¿cuánto es 9/3?", y
 60 IF y = 3 THEN GOSUB 100
 70 IF y <> 3 THEN GOSUB 200
 80 STOP
100 PRINT "BIEN": RETURN
200 PRINT "MAL": RETURN
```

Hay varias cosas nuevas en este programa, pero escríbalo exactamente y ejecútelo para ver qué pasa cuando usted contesta correctamente a la pregunta "¿cuánto es 3 × 6?" y cuando contesta mal.

Concentrémonos en los GOSUBs. Como puede ver, GOSUB 100 es la subrutina de la "respuesta correcta" (GOSUB significa "ve a la SUBrutina"), y GOSUB 200 es la subrutina de la respuesta mala. No cuesta mucho de ver que hubiera sido bastante difícil escribir este programa usando GOTOs en vez de GOSUBs, sino lo cree inténtelo. Volveremos a los **operadores lógicos** más tarde, pero ahora necesita saber que el símbolo "< >" significa "es diferente de" o bien "no igual a"; aquí, se usa para ver si las respuestas son correctas comprobando si "x" e "y" valen 18 y 3 respectivamente. A propósito, observe que GOSUB es uti-

lizado como parte de una **sentencia condicional,** un uso muy frecuente de esta particular instrucción. Las sentencias IF... THEN pueden tomar muchas formas:

```
IF A = B THEN LET A = 20
IF S$ = "para" THEN PRINT "parado"
IF a > 6 THEN RUN ('>' significa "mayor que")
```

Y más aun, de hecho, se pueden poner ecuaciones enteras en este tipo de sentencias. Intente escribir el programa anterior de un modo distinto.

LET

Ya nos encontramos con la instrucción LET en aquel programa que convertía temperaturas, y vimos que se utilizaba para crear una nueva variable. Vuelva a él un momento y échele una rápida mirada. En este caso se creó una nueva variable C que era el resultado de una operación matemática en la que entraba la variable F. Se pueden crear variables mucho más sencillas. Observe cómo se usa LET de distintas maneras:

```
10LET A = 24
20 LET N$ = "FRED"
300 LET K  =  M + 34
26 LET T = T + 1
9999 Let A$ = B$ + C$
```

En todos los casos LET hace lo mismo, si pudiéramos decirlo en palabras sería algo así como "coloca en... el resultado de...". Diciéndolo así, queda mucho más claro que decir "haz... igual a...". Esto puede parecer complicado, sobre todo al ver el ejemplo de la línea 26. ¿Cómo puede ser T igual a ella misma más 1? La respuesta es que LET crea (o vuelve a crear) una variable, la que está en la izquierda, con el resultado de las operaciones realizadas en la parte derecha del "=".

El ejemplo de la lína 26 es una manera muy común de incrementar una variable. Lo que hace es reemplazar T por lo que valía T más 1.

El ejemplo dado en la línea 10 consiste simplemente en asignar un número a una *variable numérica.* En tales casos la frase correcta para LET sería: "La variable... toma el valor de..." Algo semejante ocurre en el ejemplo siguiente con una variable de cadena, N$ que toma el valor "FRED". También se puede crear una nueva variable dependiendo de una, ya conocida (vea la línea 300); y asimismo se puede crear una nueva con la suma de dos, ya conocidas. En la línea 9999, se muestra un ejemplo de lo que llamaremos **concatenación.** Que significa que se unen dos cadenas para formar otra más larga. Veamos un ejemplo de esto:

```
10 LET B$ = "HI THERE"
20 LET C$ = "∧MAUD"
30 LET A$ = B$ + C$
40 PRINT A$
```

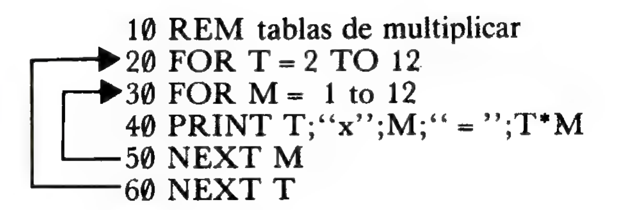
La línea 10 hace que la variable B$ tome el valor "HI THERE", la línea 20 hace C$ igual a " MAUD" (¡atento al espacio!), y la línea 30 hace A$ igual a la suma de las otras dos. Al contrario que con los números, la suma de cadenas es la unión literal de las dos partes. Si escribe este programa y lo ejecuta, obtendrá:

```
HI THERE MAUD
```

La concatenación es muy útil a veces, y volveremos a ella muchas veces a lo largo del libro.

BUCLES DENTRO DE BUCLES

Hemos visto cómo usar los bucles FOR NEXT para repetir una cosa un número determinado de veces. Hay ocasiones en las que es necesario poner bucles dentro de otros bucles. Un ejemplo muy claro es cuando queremos que el Spectrum escriba las tablas de multiplicar. Para cada tabla (haremos desde el 1 hasta el 12) hay que calcular los múltiplos de cada número, por lo tanto necesitamos poner otro bucle. Aquí tiene el programa de las tablas de multiplicar:



Observe que el bucle interior debe ser completado a cada paso del bucle exterior. No sería bueno intercambiar las líneas 50 y 60.

CONJUNTOS

Un uso muy bueno para los bucles FOR NEXT consiste en asignar valores a los conjuntos (vectores y matrices para los matemáticos). Aquí, un conjunto es como una lista de variables puestas todas en un mismo paquete, por lo que reciben el mismo nombre, y se distinguen entre sí por uno o varios números que les siguen y que se llaman subíndices. Por ejemplo, en vez de tener las variables A, B, C y D se puede crear un conjunto A (4). Para hacerlo hay que usar el comando **DIM**, que es una abreviación de DIMensión, para decirle al Spectrum cuántos elementos tendrá el conjunto (o cuántas variables habrá en el cajón). Este es un típico comienzo de programa:

```
10 DIM A(20)
20 FOR N = 1 to 20
30 PRINT "Nota para el estudiante"; N; "?"
40 INPUT A(N)
50 PRINT A(N)
60 NEXT N
```

Este programa crea un conjunto llamado A de 20 elementos. Seguidamente pregunta las notas de unos estudiantes (podían haber sido las puntuaciones de un equipo de fútbol, o cualquier otra cosa), escribiendo al lado el valor de cada respuesta (línea 50).

Se puede pensar que un conjunto está hecho de diversas variables. Por ejemplo, en este caso puede pensar que el conjunto A está hecho de las variables A(1), A(2)... hasta A(20). Como puede ver a cada uno de estos valores se le asigna la nota de un estudiante. Tiene muchas ventajas el

poder crear conjuntos como éste, en lugar de tener que crear un número elevado de variables. Es importante dominar el modo de darles valores como hace el programa anterior. Otra ventaja es que un conjunto puede ser manipulado mucho más fácilmente que una serie de variables. Por ejemplo, aquí tiene un programa que le pide 5 números cualesquiera y luego los imprime en orden descendente:

```
5 DIM A(5)
10 FOR N = 1 TO 5
20 INPUT A(N)
30 NEXT N
40 FOR X = 1 TO 5
50 FOR Y = 1 TO 4
60 IF A(Y)<A(Y + 1) THEN GOSUB 200
70 NEXT Y
80 NEXT X
90 FOR J = 1 TO 5
100 PRINT A(J);"ʌ";
110 NEXT J
120 STOP
200 LET T = A(Y)
210 LET A(Y) = A(Y + 1)
220 LET A(Y + 1) = T
230 RETURN
```

Esto puede parecer un poco complicado, pero vamos a recorrerlo línea por línea. Las primeras líneas crean la variable A, dimensionada con cinco elementos y hacen un bucle para la introducción de los números. Las líneas 40 a 80 hacen la comprobación de si A(1) es mayor que A(2), A(2) mayor que A(3), etc. Si algún número no sigue el orden descendente entonces se va a la subrutina de la línea 200 donde los dos números intercambian sus puestos en el conjunto. Para ver cómo este proceso conduce a que los números queden ordenados tal como queremos, vamos a seguir dos valores a través de todo el proceso. Supongamos que A(1) es 6 y que A(2) vale 11. En la línea 60, a la primera vuelta del bucle el programa detecta que A(Y) (aquí vale 6) es menor que A(Y + 1) (que aquí vale 11), y, por lo tanto, salta a la línea 200. En esta línea, se crea lo que se llama una **variable falsa** y se le asigna el valor de A(1). Así que T = 6. Esto se hace para retener temporal-

mente el valor de A(1) para poder intercambiar los valores de A(1) y A(2). En la línea 210 A(1) toma el valor de A(2) [por lo tanto ahora A(1) vale 11], y en la línea 220 A(2) toma el antiguo valor de A(1) al hacerlo igual a T. Así se llega al RETURN de la línea 230 con los dos elementos del conjunto cambiados de lugar.

Observará que la línea 50 hace un bucle desde 1 hasta 4 y no hasta 5. ¿Por qué es correcto? Si hubiera tomado valores desde el 1 hasta el 5 llegaría un momento en que la línea 60 intentaría comparar A(5) con A(6), pero A(6) no existe, y el programa se detendría con un mensaje de error. Hablando de detenciones observe el STOP estratégicamente colocado para evitar que el programa continúe ejecutando la subrutina una vez haya terminado lo anterior. Por último, intente averiguar usted mismo por qué únicamente se necesita recorrer el conjunto 5 veces comparando sólo los valores adyacentes para llegar a ordenarlos de mayor a menor.

Con el Spectrum también es posible construir conjuntos multidimensionales. Estos conjuntos se escriben como esto: A(3,4). En este caso, el conjunto es como un tablero de ajedrez rectangular de 3 de ancho por 4 de alto:

A(1,1)	A(2,1)	A(3,1)
A(1,2)	A(2,2)	A(3,2)
A(1,3)	A(2,3)	A(3,3)
A(1,4)	A(2,4)	A(3,4)

Como puede ver, el conjunto tiene 12 elementos, que pueden ser pensados como los nudos de una red de 3 × 4. Para ver un uso inmediato de los conjuntos de dos dimensiones piense en tres corredores que participan en cuatro carreras. La red podría ser algo como esto:

```
                      Corredores:→

     CARRERAS                  1    2    3
        ↓

                          2
                          3
                          4
```

A(1,1) sería la posición del corredor 1 en la carrera 1. A(1,2), su posición en la segunda carrera. Y así seguiríamos para todos los corredores, y todas las carreras.

También se pueden dimensionar conjuntos más grandes. Por ejemplo, el conjunto A(2,2,2,2) es perfectamente aceptable. El único límite es la cantidad de memoria de que dispone en su Spectrum. Un Spectrum de 16K no podrá tener tantos conjuntos como el de 48K. Cuando recibe un comando DIM, el Spectrum ya reserva el espacio para todos los elementos del conjunto. Así, por ejemplo, en el Spectrum de 48K usted puede entrar el comando DIM A(8000), pero un DIM A(9000) dará como resultado el mensaje de error "out of memory", que significa que se ha excedido la capacidad de la memoria.

CONJUNTOS DE CARACTERES

Del mismo modo que hemos creado conjuntos cuyos elementos son números, se pueden crear conjuntos de caracteres, palabras y frases. Lo mismo que con las variables de cadena, estos conjuntos son reconocidos por el símbolo del dólar:

```
DIM A$(30)
DIM C$(5,10)
```

Tomemos como ejemplo el conjunto C$(5,10) que es de dos dimensiones. Este conjunto puede pensarse como si estuviera hecho de 5 líneas de 10 caracteres, lo mismo que si tuviéramos 5 variables de cadena, sólo que ninguna de ellas puede exceder, en este caso, de 10 caracteres:

```
C$(1,1) C$(1,2) C$(  3)        C$(1,10)
C$(2,1) ...........            C$(2,10)



C$(5,                          C$(5,10)
```

Un conjunto de caracteres de una sola dimensión es muy similar a una variable de cadena, con la única diferencia de que debe tener una longitud fija. Esto puede ser muy

útil, por ejemplo al tomar las respuestas de un jugador en un juego. En el momento en que se dimensiona un conjunto, como A$(3), se crea una cadena A$ de tres espacios. Para probarlo, escriba lo siguiente:

```
10 DIM A$ (3)
20 PRINT " """";A$;'
```

Ejecútelo, y verá tres espacios entre las comillas, a propósito, ¿sabe por qué hay cuatro comillas a cada lado de A$? Recuerde que lo que hay en el interior de las comillas, se escribe exactamente como está.

La cadena dimensionada de longitud 1 puede usarse para aceptar la respuesta de un jugador en que un solo carácter es necesario, por ejemplo Y en lugar de YES. Recordará que si el Spectrum busca la cadena «YES», no reconoce ni la más mínima variación (incluso YES PLEASE, que significa «sí, por favor», será tomado como NO). Utilizando una variable dimensionada para asignarle la respuesta se puede solventar en parte el problema:

```
10 DIM A$ (1)
20 INPUT "¿DESEA CONTINUAR?", A$
30 IF A$ = "Y" THEN GOTO 50
40 STOP
```

Esto hace que A$ sólo contenga la primera letra de la respuesta. Por lo tanto, si la primera letra de la respuesta es una Y será considerada como YES. Esto no resuelve el problema completamente, pues puede darse una respuesta YES con un espacio delante, o bien una que empiece con Y, pero que su significado sea NO.

MANIPULACIÓN DE CADENAS

¿Se le ocurre algún truco que permita buscar en cualquier lugar de una cadena la palabra YES? Esto sería muy interesante para analizar la respuesta a la pregunta? «¿Desea continuar?». Para resolver esto fácilmente se necesitan algunos conocimientos más sobre las variables de cadena,

también llamadas string (*). Hay un gran número de funciones para el tratamiento de las cadenas. Entre ellas se encuentran **LEN** (que devuelve la longitud de la cadena) y **VAL** (que da el valor que contiene la cadena en caso de que ésta sea un número o una expresión numérica). Es la primera de estas dos, la que usaremos para buscar la palabra YES dondequiera que esté en el interior de una cadena.

Escriba este programa y ejecútelo:

```
10 INPUT A$
20 PRINT LEN A$
```

Cuando ejecute el programa usted debe poner cualquier palabra o frase e inmediatamente le aparecerá escrita en la pantalla la longitud de la cadena que ha dado como respuesta. Por ejemplo, si usted hubiera escrito «buenas noches» el resultado sería 13, que es la longitud de la cadena «buenas noches». Volvamos a nuestro problema de saber si alguien ha incluido la palabra YES en cualquier lugar de la cadena. Observe este programa, a ver si puede averiguar lo que hace. Para ello tenga en cuenta lo que acabo de decir sobre la función LEN.

```
10 PRINT "¿DESEA CONTINUAR?"
20 INPUT A$
30 FOR X = 1 TO (LEN A$ — 2)
40 IF A$ (X TO X + 2) = "YES" THEN GOTO 100
50 NEXT X
60 STOP
100 PRINT "CONTINUACION"
```

Este programa incluye una nueva posibilidad del Spectrum (que es particular del BASIC del Spectrum) en la línea 40. Se trata del troceado de cadenas, y es una herramienta extremadamente útil. Permite tratar o imprimir un trozo determinado de cadena. Aquí tiene las formas que puede tener esta operación:

N.T. El vocabulario informático español, crea palabras como: "variable string", "tratamiento string", etc. que son muy usadas.

```
A$ (1 TO 4)
A$ (1 TO)
A$ (TO 5)
A$ (5 TO)
```

Supongamos que la cadena A$ tiene longitud diez, es decir, se trata de un string de 10 caracteres. Entonces, con el primer ejemplo, se designan los 4 primeros caracteres del string. El segundo, designa a la cadena entera (desde el 1 hasta el final). El tercero, sólo tratará los cinco primeros caracteres. Mientras que el cuarto ejemplo se refiere a los seis últimos (del 5 al 10 ambos inclusive). Para verlo en acción escriba lo siguiente:

```
10 INPUT A$
20 PRINT A$(1 TO)
30 PRINT A$ (1 TO 4)
40 PRINT A$(TO 5)
50 PRINT A$ (5 TO)
```

Veamos de nuevo el programa que nos permitía detectar la palabra YES en una respuesta. Como puede ver, la línea 40 actúa de modo que toma grupos de tres letras adyacentes y los compara con la cadena «YES». Si uno de estos grupos de caracteres adyacentes es «Y», «E» y «S» entonces salta a la línea 100, donde continuará el programa, en caso de que termine el bucle sin que esto suceda, el programa se detiene en la línea 60. Observe que el límite al que puede llegar la variable de control del bucle (X), viene determinado por la longitud de la cadena A$, que se calcula mediante la función LEN.

Esta habilidad de poder manipular las cadenas puede tener otros muchos usos, aquí sólo daremos unos cuantos ejemplos, pero con un poco de imaginación, se puede llegar a hacer cosas bastante sorprendentes. Con un programa bastante más complicado, se puede conseguir que el Spectrum examine una frase hecha de un determinado vocabulario, y le conteste después.

El ejemplo que doy a continuación, no es nada complicado, le pide que introduzca dos frases y luego le dice cuál de ellas es más larga.

```
10 REM COMPARACION DE FRASES
20 INPUT "FRASE 1", X$
30 INPUT "FRASE 2", Y$
40 IF LEN X$>LEN Y$ THEN GOTO 70
50 IF LEN Y$>LEN X$ THEN GOTO 80
60 PRINT "¡SON IGUALES!": STOP
70 PRINT "LA PRIMERA ES MAS LARGA" STOP
80 PRINT "LA SEGUNDA ES MAS LARGA : STOP
```

Observe que LEN también cuenta los espacios. Puede comprobarlo comparando «Buenasnoches» con «Buenas noches».

Este nuevo ejemplo ya es un poco más complicado, ya que compara y ordena tres cadenas en lugar de dos:

```
 10 REM ORDENACION DE PALABRAS
 20 PRINT "ESCRIBA 3 PALABRAS"
 40 INPUT A$, B$, C$
 50 FOR T = 1 TO 3
 60 IF LEN A$ > LEN B$ THEN GOSUB 200
 70 IF LEN B$ > LEN C$ THEN GOSUB 300
 80 NEXT T
 90 PRINT A$: PRINT B$: PRINT C$
100 STOP
200 LET X$ = A$: LET A$ = B$: LE  B$ = X$
210 RETURN
300 LET Y$ = B$: LET B$ = C$: LE  C$ = Y$
310 RETURN
```

Primero de todo, observe cómo en la línea 40 se introducen a la vez tres variables string en un solo comando INPUT. Todo lo que tiene que hacer es separar las variables mediante coma o punto y coma. Si usa el punto y coma, las comillas de la pregunta aparecerán una al lado de la otra, mientras que si usa las comas entonces aparecerán una en cada parte de pantalla. Esto es mucho más fácil que hacer:

```
INPUT A$: INPUT B$: INPUT C$
```

La forma en que trabaja el programa es muy simple, y puede ser ampliado fácilmente para poder hacer frente a problemas más complicados. Las líneas 50 a 80 comparan dos cadenas de las tres que se introducen, si la primera es

mayor que la segunda, entonces se intercambian si no se compara la segunda con la tercera y esto se hace tres veces, hasta que quedan ordenadas de menor longitud a mayor. Los intercambios se hacen de manera semejante al programa que ordenaba números, es decir, utilizando una variable falsa.

Un uso mucho más simple de lo que hemos visto nos lo muestra el siguiente programa:

```
10 INPUT "DIME TU NOMBRE ENTERO", N$
20 FOR X = 0 TO LEN N$
30 PRINT N$ (1 TO LEN N$ — X)
40 NEXT X
```

Que produce cosas como ésta:

```
Fred Bloggs
Fred Blogg
Fred Blog
Fred Blo
Fred Bl
Fred B
Fred
Fre
Fr
F
```

También pueden leerse palabras al revés:

```
10 REM INVERSOR DE PALABRAS
20 INPUT "PALABRA", W$
30 FOR J = LEN W$ to 1 STEP — 1
40 PRINT W$ (J):
50 NEXT J
```

Encontrará divertido ver cómo se escriben nombres al revés, o intentar encontrar palabras que se lean igual del derecho que del revés.

También puede desarrollar todo esto para ayudarle a resolver rompecabezas. Por ejemplo, uno muy común, consiste en averiguar cuántas palabras de tres letras se pueden sacar de otra más grande, o colocar una palabra dentro de otra. Para hacer todo esto, le irá bien saber cómo se

puede conseguir que el Spectrum produzca números aleatorios. Esto se consigue con la función **RND.**

NÚMEROS ALEATORIOS

Los números aleatorios, son el alma de muchos juegos, tal como se ha dicho arriba también pueden ayudar a resolver otros problemas. RND produce números entre 0 y 0,999999999. Intente escribir PRINT RND varias veces (seguido de ENTER) y verá lo que quiero decir. RND se obtiene pulsando la tecla T con el cursor en modo E.

Como puede ver, esto produce números con 10 cifras decimales. Tal como está no es muy útil, pero se puede emplear la función **INT,** para conseguir números ENTEROS. INT es una abreviación de la palabra inglesa INTEGER, que significa entero. La función INT, aplicada a un número decimal, da como resultado los dígitos que hay antes de la coma. Por ejemplo:

```
PRINT INT 6,4356752346 da 6
PRINT INT 234,987987876 da 234
```

Usando el programa siguiente, se obtienen números entre 1 y 6 (por ejemplo, para usarlos como si se tratara de un dado):

```
10 REM TIRAR DADOS
15 DIM Q$(1)
20 LET N = INT (6* RND) + 1
30 PRINT N;“,”;
40 INPUT “¿OTRA TIRADA?”, Q$
50 IF Q$ = “S” THEN GOTO 20
60 STOP
```

Multiplicar RND por un número determinado, devuelve valores entre 0 y dicho número. Así, 6*RND produce números aleatorios, entre 0 y 5,999999999. Para obtener números entre 1 y 6 le añadimos 1 a 6*RND. Finalmente, para generar números enteros entre 1 y 6 le aplicamos la función INT que elimina los decimales. Dado que 6*RND + 1, produce números entre 1,0 y 6,99999999, usando INT se reducen a números entre 1 y 6, no entre 1 y 7 como

usted podría esperar. Recuerde que INT no redondea los números al entero más próximo, sino que simplemente ignora la parte decimal.

Con estos resultados, sabemos lo suficiente para intentar hacer un programa que busque palabras de tres letras que se pueden hacer con las letras de otra más grande. Inténtelo, tomando como base este programa que aquí propongo.

```
10 REM BUSCA PALABRAS
20 INPUT "PALABRA LARGA", W$
30 LET X = INT ((LEN W$*RND) + 1)
40 LET Y  =  INT((LEN W$*RND) + 1)
45 IF Y = X THEN GOTO 40
50 LET Z = INT ((LEN W$*RND) + 1)
55 IF Z = Y OR Z = X THEN GOTO 50
60 PRINT W$ (X) ;W$(Y);W$(Z)
70 GOTO 30
```

Ante todo observe que las líneas 30, 40 y 50 son muy similares. Le será muy útil utilizar EDIT para cambiar la línea 30, de modo que se transforme en la 40, ya que es mucho más rápido que escribir la línea 40 de nuevo. Lo mismo se puede aplicar para obtener la línea 50.

Todo lo que hace el programa en las líneas 3, 4 y 5, es encontrar tres números aleatorios diferentes entre sí (esto se controla en las líneas 45 y 55) y luego usarlos para escribir tres letras de la palabra larga. Las líneas 30, 40 y 50 también aseguran que el número hallado, se encuentre entre 1 y la longitud máxima de la palabra, para ello, se usa la función LEN.

Una innovación, se encuentra en la línea 55. Aquí, he usado un operador lógico llamado **OR** (que corresponde en español a la conjunción disyuntiva "o"). Esto simplemente hace lo que cabe esperar. Permite al Spectrum volver a la línea 50, tanto si Z es igual a Y *o* si Z es igual a X. Volveremos pronto a hablar de los operadores lógicos.

Habrá podido observar fácilmente, cómo se puede ir reordenando una frase sin parar para encontrar palabras en ella.

Si lo que reordenamos no es una palabra sino una frase, habrá que decirle al programa que ignore los espacios. No quiero ahora alargar este programa, pero le voy a dar unos consejos para que lo intente usted mismo. Esencial-

mente, lo que se pretende es generar números aleatorios, que vayan desde 1 hasta la longitud de la frase o palabra (restando los espacios). Luego, debe asegurarse de que cada número aparece sólo una vez. Esto se consigue generando números y comprobando constantemente si ya han aparecido. Cuando se ha generado un número de ellos igual a "LEN W$", entonces ya tenemos suficientes. Lo único que se necesita ahora es escribirlos en el orden aleatorio que ha surgido del proceso y volver a empezar de nuevo para efectuar otro intento.

Como ejemplo final sobre manipulación de cadenas, aquí tiene una para ver del popular juego del "Ahorcado", pero sin gráficos. Contiene diferentes comandos y funciones que no se han visto aún, pero intente escribirlo para ver lo que hace.

```
   5 REM EL AHORCADO
  10 DIM Q$(1)
  15 CLS
  20 LET X=INT (5*RND)+2
  30 RESTORE 100*X
  40 READ W$
  45 PRINT AT 10,5;
  50 FOR Y=1 TO LEN W$
  60 PRINT "-";" ";
  70 NEXT Y
  75 LET T=0
  80 INPUT "PRUEBA !";L$
  81 LET T=T+1: IF T=LEN W$+4 THEN  GO T
O 1000
  85 IF L$=W$ THEN  GO TO 2000
  90 FOR Z=1 TO LEN W$
 100 IF L$=W$(Z) THEN  GO TO 130
 120 NEXT Z: PRINT AT 2,5;"LO SIENTO, NO
": GO TO 80
 130 PRINT AT 10,3+(2*Z);L$
 140 GO TO 80
 200 DATA "CAMPO"
 300 DATA "MUELA"
 400 DATA "RATON"
 500 DATA "LOGICA"
 600 DATA "BIDON"
1000 REM PERDER
1010 PRINT "LO SIENTO, PERDISTE"
```

```
1020 PRINT "ERA ";W$
1030 INPUT "SIGUES ?";Q$
1040 IF Q$="S" THEN  GO TO 10
1050 STOP
2000 REM GANAR
2010 PRINT "SI ES ";W$
2020 INPUT "OTRA ?",Q$
2030 IF Q$="S" THEN  GO TO 10
2040 STOP
```

Lo que le será más difícil de entender es la parte en que aparecen los comandos READ, DATA y RESTORE. Juntos, proporcionan una gran facilidad para la obtención de información.

READ, DATA Y RESTORE

Existen tres funciones relacionadas que le permiten al Spectrum almacenar datos (en inglés, **DATA**) en una parte separada del programa, después se leen (en inglés, **READ**) los datos y se pueden asignar a conjuntos o bien a variables simples con gran facilidad y versatilidad. Aquí hay un ejemplo en el que se usan los comandos READ y DATA:

```
10 DIM A(10)
20 FOR X = 1 TO 10
30 READ A(X)
40 NEXT X:STOP
60 DATA 23,4,53,8,65,2,0,231,5,0
```

Este programa asigna el número 23 a la variable A(1), 4 a A(2), 53 a A(3), etc. En la línea 30 el comando READ le dice al Spectrum que busque la primera línea que empiece por DATA (en este caso la 60) y que lea de ella el primer trozo, este primer trozo, es asignado a la variable A(1). Cuando el bucle FOR NEXT da una nueva vuelta, y se carga a la variable de control X con su próximo valor, se encuentra de nuevo con la instrucción READ A(X). Esta vez, el siguiente dato de la línea DATA es asignado a la variable A(2), y así continuamente. Cada vez que el programa pasa por la línea READ el Spectrum incrementa en uno un contador interno para controlar qué trozo o ítem

de datos es el próximo en la lista DATA. Las líneas DATA, se pueden colocar en cualquier parte del programa, pero es aconsejable colocarlas junto a la instrucción READ o bien marcarlas con una sentencia REM por razones de claridad.

RESTORE, es la otra palabra que va con READ y DATA y hace que el contador interno vuelva de nuevo al principio de la línea DATA. Aquí hay un ejemplo en el que se usa RESTORE:

```
  5 REM READ A$
 10 DIM A$(12,5)
 20 FOR X= 1 TO 12
 40 READ A$(x)
 50 IF X/4 = INT (X/4) THEN RESTORE
 60 NEXT X
 70 REM PRINT A$
 80 FOR X = 1 TO 12
 90 PRINT A$(X);" ";
100 NEXT X
200 DATA "FRED", "BERT", "MARY", "JANE"
```

Como puede observar, los distintos datos, se separan por comas en la línea DATA, y pueden ser tanto números como cadenas de caracteres. Pero recuerde que la instrucción READ, debe leer datos numéricos, sobre variables o conjuntos numéricos, y datos string o alfanuméricos, sobre cadenas. Esto no puede ser de otra manera, y obtendrá el mensaje "Nosense in BASIC", indicándole el error. Cuando ejecute el programa obtendrá:

```
fred bert mary jane fred be
rt mary jane fred bert mary
jane
```

Como puede ver cada dato se lee tres veces sobre el conjunto A$ que contiene doce cadenas de 5 caracteres cada una. Cada vez que se cumple la condición de la línea 50 es que se ha terminado de leer todos los campos de DATA (esto ocurre cada 4 READs), y el contador interno se vuelve a colocar marcando al primer dato ("fred").

RESTORE también puede llevar un número detrás, que le indica el número de línea en que se encuentra DATA. Aquí tenemos un ejemplo:

```
 10 RESTORE 100
 20 READ X
 30 RESTORE 200
 40 READ Y
 50 RESTORE 300
 60 READ z
 70 PRINT X;",";Y;",";Z
 80 STOP
100 DATA 23,43,54
200 DATA 1,0,3
300 DATA 200,300,400
```

Una vez ejecutado se obtiene:

23,1,200

Observe cómo los otros datos de las líneas 100, 200 y 300 son ignorados completamente. Para ver más claro el efecto que produce el hecho de colocar un número de línea detrás de RESTORE, pruebe a borrar las líneas 10, 30 y 50, y vea lo que ocurre.

Ahora ya tiene los conocimientos suficientes para ver cómo actuaban READ, DATA y RESTORE en el programa del Ahorcado. Las palabras a adivinar estaban almacenadas como cadenas en las líneas DATA (de la 200 a la 600) y la línea a la que leía READ se determinaba aleatoriamente mediante la instrucción RESTORE 100 X en la línea 30 (siendo X un número aleatorio entre 2 y 6). Para añadir nuevas palabras para adivinar, sólo tiene que cambiar la línea 20, aumentando el rango de los números aleatorios X que se generan (hasta 7 para añadir una palabra más, hasta 8 para otra, etc.). La nueva palabra debe ser colocada entre comillas detrás de una instrucción DATA en la línea 700, la otra en la 800, etc. Para aumentar la dificultad, puede hacer que un amigo o familiar introduzca las palabras de manera que usted no las vea. También puede alterar la línea 81 para que el programa varíe el número de intentos permitidos. En este momento permite dos intentos más que el número de letras que tiene la palabra.

La línea DATA puede situarse en cualquier parte del programa, incluso antes de la instrucción READ que la ha de leer. Si hay más líneas DATA de las que tiene que leer el programa no ocurre nada. pero si en cambio se colocan

más READs que DATAs obtendrá un mensaje de error indicándole en qué línea READ se han excedido los datos.

Existe un uso para READ y DATA, que una vez aprendido le servirá para crear los **caracteres definidos** por el **usuario,** que se examinan en el próximo capítulo.

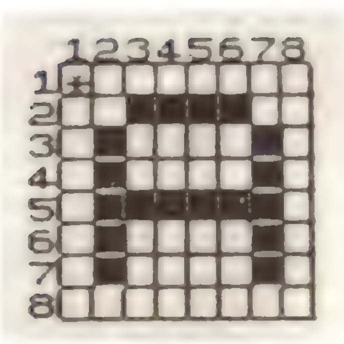
CAPÍTULO CUARTO

Los gráficos

CARACTERES DEFINIDOS POR EL USUARIO

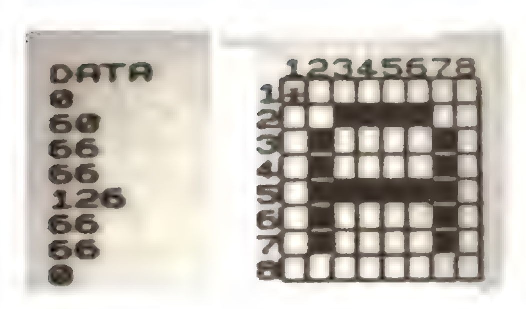
Ahora ya tiene una idea de los distintos caracteres, letras, números y símbolos que producen las teclas del Spectrum. Pero también es posible crear sus propios caracteres. Cosas como alfabetos extranjeros, invasores del espacio, símbolos técnicos, etc.

El Spectrum da gran facilidad para la creación de estos caracteres, y tiene un área de la memoria suficiente para el almacenamiento de hasta 21 caracteres. Cuando se conecta el Spectrum estos caracteres son las letras mayúsculas de la A a la U, y se obtienen pulsando dichas teclas, estando el cursor en modo G (CAPS SHIFT y 9). Para entender cómo se crea uno de estos nuevos caracteres, mire este dibujo que representa un carácter de los normales contenidos en el Spectrum.



Aquí está la letra mayúscula A tal como la hace el Spectrum. La he dibujado a propósito en un cuadrado de

8 × 8, porque cada letra, o símbolo del Spectrum está diseñado de este modo, con cada uno de los 64 cuadritos lleno o vacío. Para la creación de caracteres, usted debe pensar en este cuadrado. En el hay ocho filas de ocho puntos cada una, cada fila tiene un número asociado llamado "byte" (¿recuerda esta palabra del capítulo 1?). Aquí aparece el cuadrado con el carácter y el byte asociado a cada fila. Veamos cómo actúan estos números.



Para entender de dónde vienen estos números es necesario saber un poco de **aritmética binaria.** La aritmética binaria, consiste simplemente en contar de dos en dos en lugar de en decenas como hacemos normalmente. El problema está en que estamos acostumbrados a contar en decenas y entonces es fácil olvidar que puede haber otras maneras de hacerlo. Observemos este número escrito tal como lo hacemos normalmente:

1 0 6 4

Ahora pensemos en él como si fuera un 1 en la columna de los millares, un 0 en la de las centenas, 6 en las decenas, y 4 en las unidades. De esta manera podemos marcar cada columna del siguiente modo:

10^3	10^2	10^1	10^0
1	0	6	4

Si no está acostumbrado a contar en potencias, no se preocupe, esto, lo único que significa es que la primera columna es 1000, la segunda 100, la tercera 10 y la última 1. Ahora bien, también es posible contar en pares, y es de este modo como opera la aritmética binaria. Veamos las columnas en binario:

2^3 2^2 2^1 2^0

Las columnas son 2 al cubo (8), 2 al cuadrado (4), dos elevado a 1 (2) y 2 elevado a cero (1).

Al contar en decenas, cada columna puede tener uno de los diez dígitos (0 a 9), y de manera similar, contando en binario, cada columna puede contener uno de los dígitos (cero o uno). Ahora recuerde del primer capítulo que el Spectrum trataba los dos niveles de voltaje como 0 o 1. De esto se deduce que la aritmética binaria es fundamental en los ordenadores. Vamos a escribir un número en binario, los números 1, 2, 4 y 8 son muy fáciles:

Decimal	Binario
1	0001
2	0010
4	0100
8	1000

En el primero no hay ni ochos ni cuatro ni doses, pero sí un uno y así ocurre con los demás. ¿Cómo se escribirá el cero?, ¿y el 12? Intente convertir algunos números de binario a decimal, pero no se preocupe porque en seguida veremos una manera muy fácil de hacerlo.

Antes, sólo hemos marcado la cabecera de las cuatro columnas de la derecha, pero en binario; las cuatro de la izquierda se encabezarían 128, 64, 32 y 16 respectivamente de derecha a izquierda.

Volvamos a las filas de la letra A que estábamos viendo antes. La primera está vacía, y no es difícil adivinar que el número que le corresponde es el cero (00000000 en binario). Pero la segunda tiene llenos la mitad de sus puntos.

128 64 32 16 8 4 2 1

Para calcular a qué número corresponde esta fila necesitamos escribirla como un número binario, de manera que los puntos vacíos sean ceros, y los llenos unos. Así esta fila corresponde al número 00111100. Para convertir este número a decimal, hay que hacer los siguientes cálculos:

128*0 + 64*0 + 32*1 + 16*1 + 8*1 + 4*1 + 2*0 + 1*0

que dan como resultado

$$32 + 16 + 8 + 4 = 60$$

Mire a ver si puede averiguar cómo se obtienen los números correspondientes a las otras filas.

Ahora ya ha visto los principios de la aritmética binaria, y de cómo se relaciona con los caracteres. Voy a explicarle una manera muy sencilla de convertir números de binario a decimal:

CONSEJO

En el Spectrum hay un comando especial llamado **BIN.** Ha sido incluido para permitir la entrada de números binarios en el Spectrum. Pero también permite su conversión a la forma decimal simplemente escribiendo "PRINT BIN...".

Escriba esto para ver lo que quiero decir

```
PRINT BIN 00111100    (ENTER
PRINT BIN 01010101
PRINT BIN 11111111
```

Como recordará, el primer ejemplo, corresponde a la segunda línea de la letra A. Y el Spectrum ha imprimido el número 60 en la pantalla.

CREACIÓN DE UN CARÁCTER

Para hacerlo, el Spectrum dispone de unos comandos especiales, que le permiten introducir los ocho bytes necesarios para cada nuevo carácter. Con estos comandos, los bytes se sitúan automáticamente en la posición de memoria correspondiente a cada carácter, para que usted no tenga que efectuar ningún cálculo. Aquí viene un ejemplo en el que se crea la A mayúscula en el lugar en que se encontraba la U en modo gráfico (cursor G). Esto no es muy útil en principio, puesto que la letra A ya existe en el Spectrum, pero viendo como se construye comprenderá la forma de crear cualquier otro carácter.

```
10 POKE USR "U",0
20 POKE USR "U" + 1,60
30 POKE USR "U" + 2,66
40 POKE USR "U" + 3,66
50 POKE USR "U" + 4,126
60 POKE USR "U" + 5,66
70 POKE USR "U" + 6,66
80 POKE USR "U" + 7,0
```

Este programa puede parecerle un poco extraño, pero ahora le explicaré el funcionamiento. El comando **USR** se obtiene pulsando la tecla L con el cursor en modo E. USR "U" es la dirección de memoria en la que se almacena el primer byte del carácter definible por el USuaRio que corresponde a la letra U en modo G. Puede ver a qué dirección corresponde simplemente escribiendo:

```
PRINT USR "U"
```

Observe cómo la dirección que se obtiene concuerda con los límites del mapa de memoria que se encuentra en la página 165 del manual.

POKE simplemente significa "coloca el número que sigue en la dirección que se indica" (primero se escribe la dirección, luego una coma y por último el valor decimal del byte que se desea colocar en esa dirección). Puede imaginarse que la memoria del ordenador es como una larga fila de cajas, cada una de las cuales corresponde a una posición (byte) de memoria. La posición (o dirección) más baja es la cero, y la más alta la 65536. En cada una de estas cajas hay un número (de cero a 255, ya que es lo máximo que se puede conseguir con un byte), que de hecho es un byte de información o de datos. Así, en las direcciones que van desde USR "U" hasta USR "U" + 7 hay ocho bytes que corresponden al carácter definido por el usuario de la tecla U (que al enchufar el Spectrum corresponden, a los de la letra U, pero después de correr el programa anterior, serán los de la letra A). Ahora escriba el programa colocando las comas en su lugar y ejecútelo. Luego, con el cursor en modo G pulse la letra U y en lugar de obtener la U mayúscula como ocurría antes, aparecerá una A mayúscula. Una vez visto esto ya es muy fácil colocar cualquier otro carácter en esta tecla.

Probablemente, la mejor manera de definir nuevos caracteres, es utilizar READ DATA y RESTORE, colocando los bytes de los caracteres en líneas DATA. De todas maneras, el uso de BIN puede hacer mucho más fácil la comprensión del programa listado, ya que colocando los bytes en binario, se entrevé el carácter a definir. Por ejemplo, hagámoslo de nuevo con nuestra A mayúscula:

```
10 POKE USR"U"              BIN 00000000
20 POKE USR"U" + 1,         BIN 00111100
30 POKE USR"U" + 2,         BIN 01000010
40 POKE USR"U" + 3,         BIN 01000010
50 POKE USR"U" + 4,         BIN 01111110
60 POKE USR"U" + 5,         BIN 01000010
70 POKE USR"U" + 6,         BIN 01000010
80 POKE USR"U" + 7,         BIN 00000000
```

Esta habilidad que tiene el Spectrum de introducir los números directamente en binario, da mucha facilidad a la hora de crear un carácter gráfico.

READ Y DATA PARA LOS NUEVOS CARACTERES

Quizá la manera más elegante de crear nuevos caracteres (una vez conocidos los bytes que lo componen) es usar READ y DATA. Colocando estamentos REM que clarifiquen lo que contiene cada línea DATA, se consigue una mayor claridad. Aquí hay un ejemplo que podría muy bien ser una parte de un programa de invasores:

```
10 REM INVASORES
20 REM CREACION DEL GRAFICO
30 GOSUB 1000
40 STOP
1000 FOR A = 1 TO 7: READ X: POKE
USR"A" + A,X:
NEXT A
```

```
11ØØ DATA 24,6Ø,9Ø,126,24,36,9Ø
12ØØ RETURN
```

MOVIMIENTO

Una vez creado el carácter del invasor del espacio, ¿cómo podemos hacer que se mueva por la pantalla? Hay varias maneras de conseguirlo, pero la más simple consiste en utilizar PRINT AT. Es decir, se imprime un carácter en una posición de la pantalla, luego se borra y se imprime de nuevo una posición más adelante. Esto se puede repetir hasta que se termine la pantalla. Aquí tenemos una manera de hacerlo:

```
1Ø REM MOVIMIENTO
2Ø FOR A = Ø TO 31
3Ø PRINT AT 5,a;"*";AT 5,a;"ʌ"
4Ø NEXT A
```

Si lo prueba, probablemente se sorprenderá de lo rápido que se mueve el asterisco por la pantalla. Para aminorar la velocidad, podemos utilizar la instrucción **PAUSE.** PAUSE se usa seguida de un número positivo, para su orientación PAUSE 5Ø detiene el programa durante un segundo. Intentemos añadir PAUSE 1Ø a nuestra rutina. Escriba esta línea:

```
3Ø PRINT AT 5,a; "*": PAUSE 1Ø: PRINT AT 5,a; "ʌ"
```

Esto provoca que el movimiento del asterisco, sea más lento que antes. Por supuesto, que usted puede aumentar la velocidad, disminuyendo el número que sigue a PAUSE. También puede cambiar el asterisco por el carácter de un invasor (¿por qué no prueba con el que hemos definido antes?).

Otra manera de hacer lo mismo sin necesidad de escribir el espacio encima del carácter cada vez, es imprimiéndolos los dos a la vez, como hace este programa:

```
1Ø FOR A = Ø TO 3Ø
2Ø PRINT AT 5,A;"ʌ*"
3Ø PAUSE 2
```

```
40 NEXT A
50 PRINT AT 5,31;"
```

De esta manera, sólo se necesita la última sentencia PRINT que imprime un blanco sobre el último asterisco de la derecha.

Si lo que queremos es mover un carácter que represente a una pelota, ¿cómo podemos detectar cuando choca contra una pared? Hay dos maneras de detectar si un objeto que se mueve por la pantalla, ha chocado con algo. Uno usa el comando **SCREEN**, y el otro usa **ATTR**. Veamos cómo actúan.

SCREEN es un comando que va seguido de dos números como PRINT AT, y representan coordenadas en la pantalla de la misma manera (SCREEN significa pantalla en inglés). Su función, es localizar un lugar de la pantalla y ver qué carácter hay allí. Pruebe este simple programa para verlo en acción:

```
10 PRINT AT 10,10;"A"
20 PRINT "SCREEN$ (10, 10) ES^";
"" """;SCREEN$(10,10);"" """
```

Si lo ejecuta obtendrá:

```
SCREEN$(10,10) IS "A"
```

Observe que se necesitan cuatro comillas para producir las que envuelven a la letra A.

Esto podemos usarlo para ver cuándo un carácter en movimiento va a chocar contra otro. Introduzca este programa tal como está aquí:

```
  10 REM PELOTA SALTARINA
  20 LET T=1
  30 LET R=1
  40 FOR A=0 TO 21
  50 PRINT AT A,31;"H";AT A,0;"H"
  60 NEXT A
  70 FOR A=0 TO 30: PRINT AT 0,A;"X";AT
21,A;"X"
```

```
  80 NEXT A
  90 LET X=RND*10: LET Y=RND*15
 100 PRINT AT X,Y;"o"
 110 PAUSE 1
 120 PRINT AT X,Y;" "
 130 IF SCREEN$ (X+T,Y+R)="H" THEN  LET
R=R*-1
 140 IF SCREEN$ (X+T,Y+R)="X" THEN  LET
T=T*-1
 150 LET X=X+T: LET Y=Y+R
 160 GO TO 100
```

Esta "bola loca" es localizada del siguiente modo: El programa dibuja las paredes de arriba y de abajo con la letra X, y las de la derecha e izquierda mediante la letra H. Entonces el programa mueve la pelota, hasta que detecta el impacto con una X o una H. Cuando esto ocurre (líneas 130 y 140) entonces se cambia la dirección de la pelota multiplicando R o T por menos uno. Usted puede mantener la trayectoria de la pelota en la pantalla simplemente suprimiendo las líneas 110 y 120.

El uso de SCREEN$ tiene un inconveniente, y es que sólo reconoce los caracteres almacenados en la ROM del Spectrum, y no lee los caracteres definidos por el usuario. Esto es una pena porque se ve claramente que SCREEN puede tener una gran utilidad para juegos en los que intervengan caracteres definidos por el usuario. De todos modos, existe una manera de solucionar el problema. Esencialmente, lo que hay que hacer es primero determinar en qué lugar se va a leer un carácter definido por el usuario, luego cambiar una *variable del sistema* de manera que SCREEN crea que la tabla de caracteres empieza 256 bytes antes de la tabla de gráficos definidos por el usuario, en vez de 256 bytes antes de la tabla de caracteres en la ROM. En el siguiente ejemplo, A y B, son las coordenadas del gráfico que se quiere detectar. En esta rutina, en lugar de almacenar el carácter, se almacena su código en la variable X, pero usted puede almacenarlo como una cadena simplemente poniendo: LET A$ = SCREEN$ (A, B).

```
 10 REM subrutina para usar SCREEN$ con
los graficos definibles
 15 LET x=CODE SCREEN$ (a,b)
```

```
 20 IF x THEN  RETURN
 25 REM 23606&23607 punto para la tabla
de CHR$
 30 POKE 23606,PEEK 23675
 35 REM 23675&23676 punteros para usar
el area de graficos definidos
 40 POKE 23607,(PEEK 23676)-256
 45 LET x=CODE SCREEN$ (a,b)+112
 50 REM cambio para CHR$
 55 POKE 23606,0
 60 POKE 23607,60
 70 RETURN
```

ATTR es una abreviación de la palabra inglesa ATTRIBUTE, y usándolo, devuelve los atributos (características) de la posición que se le indica (otra vez, por tanto, necesita ir seguido de dos coordenadas como PRINT AT). ATTR es más difícil de usar que SCREEN ya que su resultado es un número que es la suma de otros distintos, cada uno de los cuales representa una característica del carácter que se encuentra en la posición indicada; por ejemplo, nos dice de qué color es la tinta, de qué color es el papel, en el lugar en que está el carácter, etc.

El número calculado por ATTR es la suma de los siguientes:

128 si la posición está en FLASH (o si no lo está)
64 si la posición está en BRIGHT
8 multiplicado por el código de color del papel
el código de la tinta

Por ejemplo:

```
10 PRINT AT 10,10; INK 2; PAPER 6;"*"
20 PRINT ATTR (10,10)
```

Ejecute esto y obtendrá la respuesta 50. Que está compuesta de 0, porque no está en FLASH, 0 porque no está en BRIGHT, 8*6 por el color del papel (PAPER), y 2 por la tinta (INK). Esto da 0 + 0 + 48 + 2 = 50. Si usted desea que el cuadrado esté en FLASH (use EDIT para añadir FLASH 1 a la línea 10) entonces el resultado será 178 (50 más 128 por el FLASH).

Realmente el uso de ATTR requiere un poco de atención. Aun así el programa de la bola loca, podría ser reescrito usando ATTR. Lo único que habría que hacer es

cambiar el color de las paredes, es decir, poner uno diferente arriba y abajo y en los lados (entonces en lugar de Xs y Hs podrían ser cualquier gráfico definido por usted). Una vez determinado el ATTR para los dos tipos de paredes (horizontal y vertical) sólo hay que cambiar las líneas 130 y 140 por otra sentencia condicional que compruebe el ATTR en lugar de SCREEN.

Por ejemplo

```
130 IF ATTR(X + t,Y + r) = 50 THEN LET r = r* - 1
```

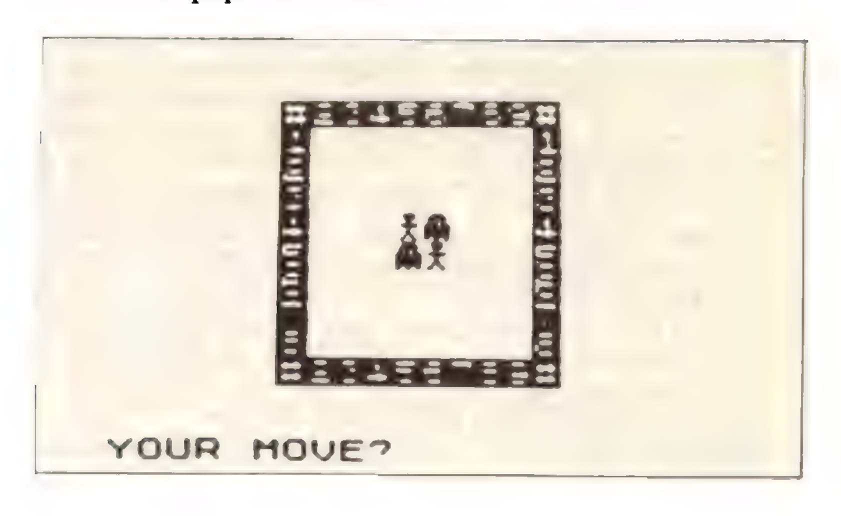
Aquí hay un programa de un juego mucho más excitante que hace uso de ATTR. Trata de unas estrellas que aparecen y desaparecen viajando por la pantalla. Las teclas 5 y 8 le mueven a derecha y a izquierda, las teclas 1 y 0 le permiten saltar diez puestos.

```
  10 REM tragon
  20 BORDER 1: PAPER 0: INK 6: CLS
  30 LET chr=480
  40 GO SUB chr
  50 LET x=5
  60 LET hit=400
  70 LET cont=1
  80 LET tantos=0
  90 LET b=0
 100 LET w=0
 110 LET s=10
 120 LET y=0
 130 LET a=INT (RND*10)
 140 FLASH 0
 150 LET b=b+2*(a>=5 AND b<28)-2*(a<4 AN
D b>0)
 160 REM a tiene que ser 'a' con modalid
ad de graficos
 170 PRINT  INK 4;AT x,y+5;"a"
 180 IF INKEY$="1" THEN  LET w=(-10 AND
s>10)
 190 IF INKEY$="0" THEN  LET w=(10 AND s
<21)
 200 IF INKEY$="5" THEN  LET w=-1
 210 IF INKEY$="8" THEN  LET w=1
```

```
 220 IF SCREEN$ (x+1,y+s)="*" THEN  GO S
UB hit
 230 LET s=s+w
 240 IF s>31 THEN  LET s=31
 250 IF s<0 THEN  LET s=0
 260 IF cont/2=INT (cont/2) THEN  GO TO
 280
 270 GO TO 290
 280 PRINT AT 20,0;TAB 8+b;"*"
 290 PRINT AT 20,0;
 300 POKE 23692,-1
 310 PRINT
 320 PRINT
 330 PRINT
 340 LET cont=cont+1
 350 PRINT AT x-1,y+s-w;" "
 360 PRINT AT x-2,y+s-w;" "
 370 IF cont=200 THEN  GO TO 460
 380 LET w=0
 390 GO TO 130
 400
 410 REM tantos
 420
 430 LET tantos=tantos+1
 440 PRINT  FLASH 1; INK 2;AT 2,12;"HIT"
 450 RETURN
 460 PRINT AT 10,5; FLASH 1;"TUS PUNTOS;
 ";tantos;" PUNTOS"
 470 STOP
 480 REM ***CHR***
 490 RESTORE 540
 500 FOR k=0 TO 7
 510 READ n
 520 POKE USR "a"+k,n
 525 NEXT k
 530 RETURN
 540 DATA 60,90,126,60,24,36,90,0
```

Aunque parezca que el uso de ATTR es menos problemático que el de SCREEN, a veces es agradable saber qué carácter se encuentra en una posición determinada. En el siguiente juego llamado COLONY SCREEN, se puede hacer que el ordenador juegue sus movimientos y que compruebe los suyos, pero esto sólo si se usan caracteres

normales, no definidos por el usuario. El juego es una versión del popular Othello.




```
   2 REM LA COLONIA
   3 BORDER 5: PAPER 0: INK 6: CLS
   4 PRINT AT 0,0;" .... .... .... ....
.... .. .. ";AT 1,0;" .... .... .... ....
 .... .. .."
   5 PRINT AT 2,0;" ..   .... ..    ..
..   ...."
   6 PRINT AT 3,0;" .... .  . ...   ..
.... ..."
   7 PRINT AT 4,0;" .... .... ...   ..
.... ..."
   8 PRINT AT 5,0;"   .. .... ..    ..
..   .... "
   9 PRINT AT 6,0;" .... .... ..    ..
.... .. .."
  10 PRINT AT 7,0;" .... .... ..    ..
.... .. ..";AT 8,30;"TM"
  11 PRINT AT 10,12; FLASH 1; PAPER 4; I
NK 6;"PRESENTA": PLOT 0,50: DRAW 15,0: D
RAW 60,120: PAUSE 500: INK 5: CLS
  12 FOR A=1 TO 15: FOR B=1 TO 6: PRINT
 FLASH 1; PAPER B; INK 9;AT 10,12;"COLON
IA": BEEP .01,RND*20: PRINT AT RND*20,RN
D*30; INK RND*6;"HC": NEXT B: NEXT A
  13 PRINT  FLASH 0; PAPER 3; INK 6;AT 4
,0;"PARA JUGAR "; FLASH 0;"ENTRE LAS COO
```

```
RDENADAS DE POSICION PONIENDO PRIMERO LA
 FILA Y LUEGO LA COLUMNA"
  14 PRINT  INK 6; PAPER 3;"EL JUEGO CON
SISTE EN APODERARSE DE LA COLONIA. CUAND
O UN EXTRATERRESTRE TIENE UN HUMANO EN C
ADA LADO, ESTE SE CONVIERTE EN HUMANO. A
LGO SIMILAR OCURRE CUANDO DOS ALIENS ROD
EAN A UN HUMANO."
  15 PRINT  INK 6; PAPER 3;'''"PULSE 'O'
PARA NO MOVERSE"''"PULSE 'S' PARA ACABAR
"''"PULSE TECLA PARA EMPEZAR"
  16 PAUSE 0: CLS
  17 LET K=0: LET L=0: LET P=0: LET H=0
  18 LET R=0: LET V=0: GO SUB 3000
  19 RESTORE 50
  20 FOR A=10 TO 19
  21 READ X$
  30 PRINT  INK 3; INVERSE 1;AT 5,A;X$
  40 NEXT A
  50 DATA "[illegible]","2","3","4","5","6","7","8
","9","[illegible]"
  60 FOR Z=1 TO 8
  65 FOR F=1 TO 2
  70 READ X
  80 PRINT  INK 3; INVERSE 1;AT 5+Z,10+(
9*V);X;
  90 LET V=V+1
 100 IF V=2 THEN  LET V=0
 105 NEXT F
 110 NEXT Z
 120 DATA 1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,
8
 125 RESTORE 50
 130 FOR A=10 TO 19
 140 READ X$
 150 PRINT  INK 3; INVERSE 1;AT 14,A;X$
 160 NEXT A
 170 PRINT AT 9,14; INK 6;"H";AT 9,15; I
NK 4;"C";AT 10,14;"C";AT 10,15; INK 6;"H
"
 175 POKE 23609,100
 180 INPUT "MUEVES?"; LINE A$
 181 IF A$="O" THEN  GO TO 8000
 182 IF A$="S" OR A$="s" THEN  GO TO 990
0
```

```
 183 LET R=0: LET P=0: LET H=1
 184 BEEP .1,1: BEEP .1,3: BEEP .1,2: BE
EP .2,4
 186 IF LEN A$<2 THEN  GO TO 180
 187 IF CODE A$(1)<49 OR CODE A$(1)>57 T
HEN  GO TO 180
 188 LET X=VAL A$(1): LET Y=VAL A$(2)
 191 IF ATTR (5+X,10+(Y-1))<>5 THEN  GO
TO 230
 200 FOR Q=-1 TO 1: FOR W=-1 TO 1
 210 IF ATTR (5+X+Q,10+(Y-1)+W)=4 THEN
GO SUB 1000
 220 NEXT W: NEXT Q
 225 IF R<>0 THEN  GO TO 240
 230 IF R=0 OR H=0 THEN  PRINT AT 3,8; F
LASH 1;"ENTRADA ILEGAL!": GO SUB 2000: P
RINT AT 3,8;"                "
 235 IF H=0 THEN  PRINT AT X+5,B;" "
 240 IF R=0 THEN  GO TO 180
 245 IF H=0 THEN  GO TO 180
 250 PRINT AT 5+X,10+(Y-1); INK 6;"H"
 260 GO TO 5000
1000 LET R=1
1010 RETURN
2000 FOR E=0 TO 5: FOR A=-10 TO -20 STEP
 -1: BEEP .01,A: NEXT A: NEXT E
2010 RETURN
3000 RESTORE 3010
3005 FOR A=0 TO 7: READ X: POKE USR "C"+
A,X: NEXT A
3010 DATA BIN 00011100,BIN 00011100,BIN
00101010,BIN 01101011,BIN 01111111,BIN 0
1111111,BIN 01101101,BIN 01001001
3020 RESTORE 3040
3030 FOR A=0 TO 7: READ X: POKE USR "H"+
A,X: NEXT A
3040 DATA BIN 00011100,BIN 00011100,BIN
00001000,BIN 00111110,BIN 00001000,BIN 0
0001000,BIN 00010100,BIN 00100010
3050 RETURN
5000 LET A=5+X: LET B=10+(Y-1)
5100 LET H=0
6000 FOR Q=-1 TO 1 STEP 1
6010 FOR W=-1 TO 1 STEP 1
```

```
6020 IF ATTR (A+Q,B+W)=4 THEN  GO SUB 70
00
6030 NEXT W: NEXT Q
6036 IF H=0 THEN  GO TO 230
6040 GO TO 8000
7000 IF ATTR (A+(2*Q),B+(2*W))=6 THEN  P
RINT  INK 6;AT (A+Q),(B+W);"H"
7010 IF ATTR (A+Q;B+W)=6 THEN  LET H=1
7020 RETURN
8000 LET C=INT (14*RND)
8005 FOR A=C TO 14
8010 FOR B=11 TO 19
8020 IF ATTR (A,B)=4 THEN  GO SUB 8500
8030 NEXT B: NEXT A
8040 LET C=6: LET P=P+1
8050 IF P=2 AND A$="O" THEN  GO TO 9900
8060 GO TO 8005
8500 FOR Q=-1 TO 1
8510 FOR W=-1 TO 1
8520 IF ATTR (A+Q,B+W)=6 THEN  GO TO 855
0
8530 NEXT W: NEXT Q
8540 RETURN
8550 IF ATTR (A+(2*Q),B+(2*W))=5 THEN  G
O TO 9000
8560 GO TO 8530
9000 BEEP .1,4: BEEP .1,2: BEEP .1,3: BE
EP .2,1
9002 BEEP .1,4: BEEP .1,2: BEEP .1,3: BE
EP .2,1
9005 PRINT  INK 4;AT A+(2*Q),B+(2*W),"C"
9010 PRINT  INK 4;AT A+Q,B+W;"C"
9020 LET A=A+(2*Q): LET B=B+(2*W)
9030 FOR Q=-1 TO 1: FOR W=-1 TO 1
9040 IF ATTR (A+Q,B+W)=6 THEN  GO TO 906
0
9050 NEXT W: NEXT Q
9054 LET T=1
9055 GO TO 180
9060 IF ATTR (A+(Q*2),B+(W*2))=4 THEN  P
RINT  INK 4;AT A+Q,B+W;"C"
9070 GO TO 9050
9900 LET H=0: LET C=0
9910 FOR A=6 TO 14: FOR B=11 TO 19
```

```
9920 IF ATTR (A,B)=6 THEN  LET H=H+1
9922 IF ATTR (A,B)=4 THEN  LET C=C+1
9925 IF A$="S" THEN  GO TO 9940
9930 IF ATTR (A,B)=0 THEN  GO TO 180
9940 NEXT B: NEXT A
9941 IF C=H THEN  PRINT AT 2,12; FLASH 1
;"DIBUJA": STOP
9950 IF C>H THEN  GO TO 9990
9960 PRINT  PAPER 6; INK 2; FLASH 1;AT 2
,4;"GANASTE CON ";H;" A MI";C
9965 FOR A=20 TO 40: BEEP .2,A: NEXT A:
GO TO 9965
9970 STOP
9990 PRINT  INK 2; PAPER 6; BRIGHT 1; FL
ASH 1;AT 2,0;"HE GANADO CON ";C;" A TUS
";H
```

Como puede observar, en este programa se usan muchas líneas del tipo:

```
10 IF CODE SCREEN$ (10,10) = 42 THEN GOTO 1000
```

El uso de **CODE** permite obtener lo que se llama el **código ASCII** de cualquier carácter producido por el Spectrum. En el apéndice se proporciona un listado de todos los códigos ASCII. Todos los caracteres que se usan tienen un número asociado a ellos. Si usted está interesado en esto, el siguiente programa le listará todos los caracteres y sus códigos.

```
10 FOR A = 32 TO 255
20 PRINT "CODE^";A "^ES^";CHR$A
30 NEXT A
```

CODE y **CHR** hacen lo inverso el uno del otro, así, CHR 122 es 'z' mientras que CODE 'z' es 122. En el programa anterior pulse ENTER como respuesta a la pregunta "scroll" para visualizar una nueva pantalla de caracteres y sus códigos. No he incluido los primeros 31 caracteres porque son códigos de control y al intentar imprimir los códigos de control, pueden producirse efectos muy peculiares.

Los códigos del 0 al 5 y del 24 al 31, no se usan en el Spectrum, los otros son:

CÓDIGO	FUNCIÓN
6	coma de PRINT
7	EDIT
8	cursor izquierda
9	cursor derecha
10	cursor abajo
11	cursor arriba
12	DELETE
13	ENTER
14	número
15	sin uso
16	INK (tinta)
17	PAPER (papel)
18	FLASH (parpadeo)
19	BRIGHT (brillo)
20	INVERSE
21	OVER
22	AT
23	TAB

Estos caracteres de control, pueden usarse como comandos en líneas de programa, o también pueden ser muy útiles para usar desde un programa, aspectos que sólo es posible utilizarlos directamente desde el teclado. De hecho, si usted quiere mover el cursor por la pantalla durante un programa, sólo puede hacerlo usando PRINT AT o bien los caracteres de control. Por ejemplo, supongamos que usted quiere subrayar la letra 'a'. Aquí tiene cómo puede hacerse esto usando el carácter de control 8 (cursor izquierda):

```
PRIN   OVER      a";CHR$ 8;
```

OVER significa que se van a fusionar los dos caracteres en lugar de que el segundo borre al primero.

La producción de color mediante el uso de los caracteres de control, será tratada más tarde, pero ahora, nos será útil observar, que los caracteres de control pueden unirse

(concatenarse) como cadenas, para formar un solo cuerpo gráfico, etiquetado con un mismo nombre:

```
10 LET A$ = CHR$17 + CHR$6 + "Hi"
20 PRINT A$
```

Esto imprime la palabra "Hi" en papel amarillo. Por supuesto que A$ puede ser mucho más larga. Este método es equivalente a que la variable A$ recurde por usted lo que tiene que imprimir en un momento dado del programa, pero sólo es eficaz si tiene que imprimir lo mismo varias veces. Aunque volveremos a usar los caracteres de control para el tratamiento del color, observe cómo se ha conseguido esto. La cadena A$ está formada de CHR$ 17 para cambiar el papel (17 es el código de PAPER), y luego se pone el código del color después del próximo CHR$. De una manera similar CHR$ 18 + CHR$ 1 provocará el parpadeo, y así sucesivamente.

DETALLES GRÁFICOS

Algunos ordenadores, tienen lo que se llama **modos de resolución.** Esto permite al usuario escoger el nivel de resolución gráfica que desee en cada momento; es decir, cómo es de grande el punto más pequeño que se puede imprimir, o lo que es lo mismo en cuántas unidades se divide la pantalla. El Spectrum sólo posee un nivel de resolución que es razonablemente alto. Se puede pensar que la pantalla está compuesta de una red de puntos de 256 de ancho por 176 de alto. Para ennegrecer uno de estos puntos, se usa el comando PLOT. El punto queda pintado con el color de la tinta. Pruebe este ejemplo:

```
10 PLOT 128,88
```



Una vez ejecutado el programa, fíjese bien y podrá observar un tenue punto en el centro de la pantalla. Lo que hace esta línea es pintar un punto en la posición 128 (puntos) empezando por la izquierda y 88 contando de abajo para arriba (así por ejemplo, la posición Ø,Ø corresponde a la esquina superior izquierda de la pantalla). Para ver actuar a la instrucción PLOT más activamente vamos a pintar más puntos en la pantalla:

```
 5 PAPER Ø:CLS
1Ø FOR X = 1 TO 2ØØ
2Ø PLOT INK 7*RND; 255*RND,
175*RND
3Ø NEXT X
```

Ejecute este programa y verá un "cielo de noche" lleno de estrellas (puntos) coloreadas. Observe cómo se usa RND en la línea 2Ø para escoger de forma aleatoria tanto el color de la tinta como la posición de los puntos, que están entre Ø y 255 y entre Ø y 175.

Es fácil modificar un poco este programa para que nos muestre realmente las habilidades del Spectrum:

```
10 INK 0: BORDER 1: CLS
20 LET A= RND*127
30 LET B = RND *87
40 PLOT INK RND*7;128 + A,88 + b: PLOT INK RND*7;
   128 - A,87 - B: PLOT INK RND*7; 128 - A,87 + B:
   PLOT INK RND *7; 128 + A,87 - B
50 GOTO 20
```

Observe atentamente la línea 40 y trate de averiguar por qué los puntos se sitúan simétricamente en la pantalla.

Al contrario de lo que dije anteriormente, el Spectrum posee otro nivel de resolución más bajo, que se consigue usando los gráficos ya construidos que se encuentran en las teclas de los números del 1 al 8. Hay un gráfico diferente en cada una de esas teclas, pero no se obtienen pulsando simplemente CAPS SHIFT. Para ver cómo son estos gráficos, coloque el cursor en modo G (CAPS SHIFT y 9). Ahora pulse las teclas del 1 al 8 para ver los primeros ocho gráficos, y luego manteniendo apretado CAPS SHIFT pulse de nuevo estas teclas para obtener otros ocho. Como puede observar los segundos son los inversos de los primeros. Mire el gráfico que se obtiene pulsando la tecla 1 tal cual, verá que es un pequeño cuadrado que ocupa exactamente una cuarta parte del cuadrado que le corresponde a un carácter. Si pensamos que este pequeño cuadrado es un punto, podemos escribir una nueva versión del programa anterior en una resolución más baja:

```
 10 BORDER 1: PAPER 0: CLS
 20 LET A = RND *10
 30 IF A>.5 THEN GOTO 60
 40 INK 1 + RND*6
 50 GOTO 70
 60 INK 0
 70 LET X= RND *10
 80 LET Y = RND *15
 90 PRINT AT 10+ X, 15 + Y; "▝"
100 PRINT AT 10 + X,15 - Y; "▝"
110 PRINT AT 10 - X,15 + Y; "▝"
120 PRINT AT 10 - X,15 - Y; "▝"
130 GOTO 20
```

Al contrario que el programa anterior, éste hace que más o menos en la mitad de ocasiones el color de la tinta sea

el mismo que el del papel (línea 30). Debido a esto, los puntos se crean y se destruyen constantemente. Por supuesto que si lo desea puede añadir esto muy fácilmente en el programa anterior.

RAPIDEZ EN LOS DIBUJOS

Su Spectrum no sólo coloca puntos en la pantalla, sino que también dibuja líneas utilizando el comando DRAW (que significa dibujo). Aquí tiene un ejemplo:

```
10 PLOT 0, 128: DRAW 255,0
```

Esto colocará una línea horizontal en la parte alta de la pantalla. Si hay otro color que no sea negro, la línea aparecerá en este color; pero también se puede definir el color de la tinta localmente. Pruebe:

```
10 PLOT 128,88
20 DRAW INK RND*7;(RND*127)*((RND*2) –
   (RND*87)*((RND*2) – 1)
30 GOTO 10
```

Por lo simple que es el programa, produce efectos muy sorprendentes. La línea 20 hace todo el trabajo. Primero asigna un valor aleatorio para la tinta, y lo mismo hace con las coordenadas de la línea que hay que dibujar (esto se hace desde el punto "PLOTeado" en la línea 10). (RND*127) y (RND*87) definen puntos a una distancia máxima de 127 horizontales y 87 verticales. Al añadir [(RND*2)-1] hacen que las coordenadas sean positivas o negativas aproximadamente en la mitad de los casos, lo que provoca que la línea dibujada llegue a derecha, izquierda, arriba o abajo del punto "PLOTeado". Si aún no lo ve claro modifique el programa para que le escriba los números que se obtienen aleatoriamente, esto le ayudará a ver cómo actúa DRAW.

CÍRCULOS

Del mismo modo que puede dibujar líneas, el Spectrum también puede dibujar círculos con un solo comando. El

comando **CIRCLE** se obtiene con el cursor en modo E pulsando simultáneamente SYMBOL SHIFT y la tecla H.

Este es un ejemplo en el que se dibuja un pequeño círculo en el centro de la pantalla, y otro concéntrico a su alrededor:

```
10 CIRCLE 128,88,10
20 CIRCLE 128,88,87
```

De nuevo la tinta con que desea dibujar el círculo se puede definir fácilmente sin ningún problema:

```
CIRCLE INK 2; 128,88,87
```

También se puede indicar localmente el papel, pero como éste sólo se define en un cuadrado entero, los efectos son un poco raros:

```
CIRCLE INK 3; PAPER 6;100,30,25
```

Los arcos se pueden obtener fácilmente añadiendo un nuevo elemento a los ya familiares PLOT y DRAW:

```
10 PLOT 128,88: DRAW 50,50,PI
```

Este programa dibuja un semicírculo empezando en el punto 128,88 y terminando en el punto situado a 50 posiciones a la derecha y 50 hacia arriba.

Si lo desea puede hacer una nueva versión del programa que dibujaba líneas, usando arcos en vez de líneas rectas, para ello simplemente tiene que reemplazar la línea 50 por ésta última que hemos visto añadiéndole los RNDs correspondientes para que todo se produzca aleatoriamente.



CAPÍTULO QUINTO

Algunos cabos sueltos

ES LÓGICO

En muchas ocasiones, nosotros queremos comparar un número con otro, o una palabra con otra. Los símbolos necesarios para tales comparaciones son: "<", ">", "< =", "> =", "< >"; que respectivamente significan: "es menor que", "mayor que", "menor o igual que", "mayor o igual que" y "distinto que". Su utilidad está muy clara.

```
10 INPUT "ESCRIBA UN NUMERO"; A
20 LET B = 100
30 IF A > B THEN PRINT "DEMASIADO GRANDE"
40 IF A < B THEN PRINT "DEMASIADO PEQUEÑO"
50 GOTO 10
```

Esto es la esencia del juego "Adivine mi número" (aunque usted podría naturalmente añadir la condición de "si A = B entonces..." o bien hacer que B sea un entero cualquiera usando RND). El uso de "menor o igual" y de "mayor o igual" tampoco presenta ningún problema. Estos signos, también se pueden usar para comparar palabras, en este último caso el orden tomado es el alfabético. Aquí tiene algunos ejemplos:

'A' es menor que 'B'
'Piedra' es mayor que 'Pera'
'ZZ' es menor que 'ZZZ'
'XY' es mayor que 'XD'

Como puede ver, si se comparan dos letras, la menor es la primera en el orden alfabético. Si las cadenas tienen más de una letra, primero se comparan entre sí las dos primeras, luego las dos segundas, y así hasta que una de ellas es mayor que la otra o bien se terminan las cadenas. Esto puede utilizarse para que la respuesta a una línea INPUT sea justamente una letra del alfabeto:

```
10 INPUT "UNA LETRA"; A$
20 IF A$< "A" OR A$> "Z" AND A$< "a" OR
   A$> "z" THEN GOTO 10
30 PRINT "ESCRIBISTE"; A$
```

He puesto la detección tanto para mayúsculas como para minúsculas, puesto que ya vimos que el Spectrum es muy exigente en este tipo de distinciones. Hay un ejemplo que demuestra claramente esto, cuando se pregunta cuál es el más pequeño de estos dos diablos:

DIABLO y diablo

La respuesta es por supuesto el primero de los dos, ya que las letras mayúsculas tienen el código más bajo que las minúsculas. Con esto se ve que las comparaciones no se efectúan con los caracteres en sí mismos, sino entre sus códigos.

OPERADORES LÓGICOS

En el programa anterior, he usado OR para decir "o si esto es verdadero". Así, la sentencia "IF A = 1 OR B = 2 OR C = 3 THEN PRINT "VERDAD" imprimirá la palabra "VERDAD" sólo si alguna de las tres cosas es verdadera. A esto, los matemáticos lo llaman disyunción. AND es la conjunción, es decir que la condición se cumple si todas sus partes son verdaderas:

IF A = 1 AND B = 2 THEN PRINT "VERDAD"

imprimirá la palabra "verdad" sí y sólo si el valor de A es 1 y el de B es 2. IF A THEN PRINT "VERDAD" imprimirá la palabra "verdad" si el valor de A no es cero. De un modo similar:

```
PRINT (CHR$ 50 AND A = 1)
```

sólo imprimirá el número 2 (CHR$ 50) si A vale 1. En caso contrario, no imprimirá nada en absoluto (ni tan sólo un espacio). Esto es muy útil cuando usted desea que pasen cosas distintas según la tecla que se pulsa o bien dependiendo del valor de una variable. Sin embargo, no es posible hacer comparaciones lógicas directamente, tales como PRINT A AND B donde A y B tienen unos valores dados. En el Spectrum, esto provoca la impresión de A si B no es 0, pero *no* efectúa una conjunción (AND) lógica de los dos números. ¿Qué es un AND lógico? Bien, consideremos los siguientes números binarios:

01010101 y 11111111

Efectuar un AND lógico de estos dos números consiste en mirarlos por columnas, dando un 1 si los dos son unos, y cero en caso contrario.

Para efectuarlo, se pone un número debajo de otro:

```
01010101
11111111
--------
01010101
```

Por lo tanto dos unos (1) dan un 1, dos ceros (0) dan un cero, y uno de cada da 0:

```
  10
 ----
1|10
0|00
```

Nosotros podemos simular esta operación (que es posible en otros ordenadores) usando las comparaciones de cadenas:

```
 5 REM DECIMAL A BINARIO
10 INPUT N
20 PRINT N
30 LET X = N
40 LET A$ = "00000000"
50 FOR Y = 1 TO 8
```

```
60 LET A$(Y) = ("1" AND X/(2↑(8 - Y))>=1)
70 IF A$(Y) = "1" THEN LET X = X - (2↑(8 - Y))
80 IF X = 0 THEN GOTO 110
90 IF A$(Y)<> "1" THEN LET A$(Y) = "0"
100 NEXT Y
110 PRINT N; "es"; A$ "EN BINARIO"
```

y:

```
 10 REM PARA REPLICAR 'Y'
 20 INPUT "A?";X
 25 PRINT "A=";X
 30 INPUT "B?";Y
 35 PRINT "B=";Y
 40 DIM A$(8): DIM B$(8)
 50 FOR T=1 TO 8
 60 LET A$(T)="1" AND (X/(2^(8-T))>=1)
 65 IF A$(T)=" " THEN  LET A$(T)="0"
 70 LET B$(T)="1" AND (Y/(2^(8-T))>=1)
 75 IF B$(T)=" " THEN  LET B$(T)="0"
 80 LET X=X-(2^(8-T) AND A$(T)="1")
 90 LET Y=Y-(2^(8-T) AND B$(T)="1")
100 NEXT T
105 PRINT "A=";
110 FOR W=1 TO 8
120 PRINT A$(W);: NEXT W
125 PRINT : PRINT
126 PRINT "B=";
130 FOR W=1 TO 8: PRINT "A Y B=";
135 DIM C$(8)
140 FOR J=1 TO 8
145 LET C$(J)="0"
150 IF A$(J)="1" AND B$(J)="1" THEN  LE
T C$(J)="1"
160 PRINT C$(J);: NEXT J
200 PRINT : PRINT "QUE ES>>";VAL ("BIN
"+C$)
```

Como puede observar las dos hileras de ceros y unos obtenidas al convertir los números decimales en binarios, son comparadas en el segundo programa. El resultado final puede ser pasado a decimal usando la función VAL, que

devuelve el valor numérico de una cadena, si ésta está compuesta de números o de variables. El complementario de VAL es STR$ que convierte cualquier número o línea de dígitos e incluso una ecuación en una cadena, y que, por lo tanto, puede ser manipulada como tal. Pero en nuestro programa no se permite la entrada directa de números en binario como ocurre aquí:

```
10 INPUT "INTRODUZCA EL PRIMER NUMERO BINARIO", N1
20 INPUT "Y EL SEGUNDO", N2
30 LET A$ = STR$ N1
40 LET B$ = STR$ N2
```

El problema es que los ceros de la izquierda serán eliminados al entrar los valores en variables numéricas.

LA IMPRESORA

La impresora Sinclair puede ser un excelente complemento de su Spectrum. Aunque en un principio fue designada para trabajar con el ZX81, también funciona razonablemente bien con el Spectrum. De todos modos, ocurre que el Spectrum la hace ir demasiado deprisa con lo que a veces hay que hacer varias copias para obtener una que esté bien. Pero no hay otro remedio si quiere tener sus programas listados.

El hecho de poseer un listado de su programa le permite revisarlo para buscar y localizar los posibles errores sin el inconveniente de ver sólo una parte al mismo tiempo. Más aún, el tener un catálogo listado de sus programas le permite recordar qué es lo que ha hecho y, quizá con un poco más de trabajo, en qué cinta y lugar los tiene registrados. El tener copias en papel puede ser útil por otras razones, desde copias de los resultados de los cálculos hasta complejos resultados científicos, pasando por el balance mensual de sus cuentas en el banco. Los comandos que hacen funcionar la impresora, son muy claros. LPRINT hace lo mismo que PRINT sólo que escribe la información en la impresora en vez de en la pantalla. LLIST trabaja como LIST, pero el listado lo hace en papel. Sin embargo, hay una ventaja, la impresora no se para y pregunta "scroll". Si sólo quiere una parte del programa listado, no habrá ningún problema dependiendo de qué parte quiera.

Si lo que quiere es el listado desde un número de línea haga LLIST n, donde n es el número de línea. Si en cambio desea una pantalla listada en papel entonces use el tercer comando para la impresora: COPY, que hace una copia en papel de toda la pantalla. Naturalmente, COPY es también muy útil para tener un recuerdo de los dibujos.

Si usted quiere listar menos de una pantalla en la mitad del programa entonces la cosa se complica un poco. Esto se puede hacer usando SCREEN$. Para hacer un listado parcial, lo primero que necesita es poner el listado en la pantalla, y luego escriba esto sin número de línea:

```
FOR A = Ø TO 31: LPRINT SCREEN$(Ø,A):NEXT A
```

Esto imprime la primera línea de la pantalla. La siguiente se hace de un modo similar cambiando el primer número de los paréntesis de SCREEN$ por un 1, y así sucesivamente. Si usted ya sabe que solamente quiere listar las cinco primeras líneas de la pantalla, entonces incluya esto en su programa:

```
 999 LIST
1ØØØ FOR A = Ø TO 4
1Ø1Ø FOR B = Ø TO 31
1Ø2Ø LPRINT SCREEN$ (A,B)
1Ø3Ø NEXT B
1Ø4Ø NEXT A
```

Imprimir sólo un trozo de la pantalla es un poco complicado. El problema está en que el Spectrum almacena los caracteres a imprimir en un área de la memoria llamada PRINTER BUFFER, y espera hasta que está llena. Tal como podía esperar esta área mide 256 bytes, es decir, ocho bytes por cada carácter de los 32 que tiene una línea. Así que la intención es de que se imprima una línea cada vez.

Las únicas ocasiones en que el Spectrum no espera a que el BUFFER esté lleno es cuando se introduce un carácter de Newline (ENTER), una coma, un TAB o una comilla, que requieren un salto de línea. Es por esta razón que no podemos definir un trozo para imprimir como éste:

```
10 FOR A = 5 TO 15
20 FOR B = 4 TO 12
30 LPRINT SCREEN$ (A,B);
40 NEXT B: NEXT A
```

Esto simplemente almacena caracteres en el BUFFER de la impresora hasta que termina el programa o se llena con 32 caracteres. Con este programa se puede solventar todo esto:

```
   5 REM LA VENTANA IMPRESA
   6 REM vamos a confeccionar un texto q
ue va a escribirse en un cuadrado defini
do por R1, R2, C1, C2
  10 INPUT "PRIMERA FILA?",R1
  20 INPUT "ULTIMA FILA?",R2
  30 INPUT "PRIMERA COLUMNA?",C1
  40 INPUT "ULTIMA COLUMNA?",C2
  50 DIM T$((R2-R1)+1,32)
  60 FOR X=R1 TO R2
  70 FOR Y=C1 TO C2
  80 LET T$(X-R1+1,Y)=SCREEN$ (X,Y)
  90 NEXT Y: NEXT X
 100 FOR A=1 TO (R2-R1)+1
 110 LPRINT T$(A)
 120 NEXT A
```

Para imprimir caracteres en inverso, simplemente escriba INVERSE 1 antes de enviarlos e INVERSE 0 después de haberlo hecho. Esto no afecta a la pantalla para nada, y no trabajará cuando se use COPY. Por ejemplo:

```
10 LET A$ = "Inverse"
20 INVERSE 1: LPRINT A$: INVERSE
0
```

O bien un listado en inverso:

```
INVERSE 1: LLIST: INVERSE 0
```

También se obtiene el mismo efecto haciendo POKE 23697, 12 en lugar de INVERSE 1 y POKE 23697, 0 en lugar de INVERSE 0.

Lo que hace POKE es simplemente poner un trozo de información directamente en una posición de la memoria del ordenador. Usted puede imaginarse que la memoria es como una torre de cajas, cada una de ellas numerada desde 0 hasta 65.000 más o menos. A cada una de estas cajas se le llama posición de memoria o dirección. Mire el dibujo de las direcciones en el apéndice, llamado Mapa de la Memoria. Para ver lo que hay en una posición determinada se usa PEEK.

CLOSE# y OPEN# también trabajan con la impresora, aunque están pensados para el tratamiento de ficheros en diskette. Estos comandos, pueden conectar y desconectar la impresora. Si usted escribe OPEN# 2 "nombre" (la impresora ha sido designada como el dispositivo 2) entonces, todos los PRINTs o LISTs que haga a partir de ahora irán a parar a la impresora en vez de dirigirse a la pantalla. CLOSE# devuelve la situación a su estado natural. Obviamente, es más fácil usar OPEN# y CLOSE# para determinar dónde debe dirigirse un trozo de información, que escribir complicadas sentencias condicionales con PRINT o LPRINT.

Por último, le puede interesar una manera de escribir caracteres en la impresora de manera que tengan el doble de altura. Esto se hace utilizando de nuevo la instrucción SCREEN$. Esta vez se usa para encontrar el código del carácter que se encuentra en una posición determinada. Sabiendo el código, el programa se va al área de la ROM donde el Spectrum almacena los bytes correspondientes a ese carácter. Allí encuentra cuáles son los ocho bytes que corresponden a ese carácter, y los coloca (POKE) en el buffer de la impresora. Sin embargo, se coloca dos veces cada byte y el resultado de esto es que primero se imprime la mitad de arriba de todos los caracteres de la línea, y luego la mitad de abajo.

```
10 REM PONER EL CARACTER DE LA POSICION 90
EN LA IMPRESORA A MODO DE EJEMPLO
20 LET code = CODE SCREEN$ (0, 0)
30 LET X = 15616 + code *8        (tabla de caracteres)
40 FOR A = 23296 TO 23552 STEP 2
                                  (buffer de la impresora)
50 POKE A, PEEK X
```

```
60 POKE A + 32, PEEK X
70 LET X = X + 1
80 NEXT A
```

Evidentemente, habrá que escoger cada posición de la pantalla. La tabla de caracteres en la ROM, va desde la posición 15616 en adelante y los bytes de cada carácter se encuentran desde la posición 15616 + 8* code hasta la posición 15616 + 8* code + 8 (donde code representa el código del carácter en cuestión). Los caracteres se almacenan en el buffer del mismo modo que se imprimen en la pantalla, primero se pone el primer byte de los 32 caracteres, luego el segundo, etc. En la línea 60 del ejemplo se "POKEa" el mismo byte dos veces en el buffer unas 32 posiciones después de la última.

FUNCIONES

La última función que nos ofrece el Spectrum es DEF FN acompañada de FN. Estos dos comandos son muy potentes, ya que permiten definir una ecuación compleja con DEF FN A(xxx) y usarla en el programa simplemente llamándola mediante FN A(xxx). Por ejemplo, usted puede definir una función que sea el cubo de un número:

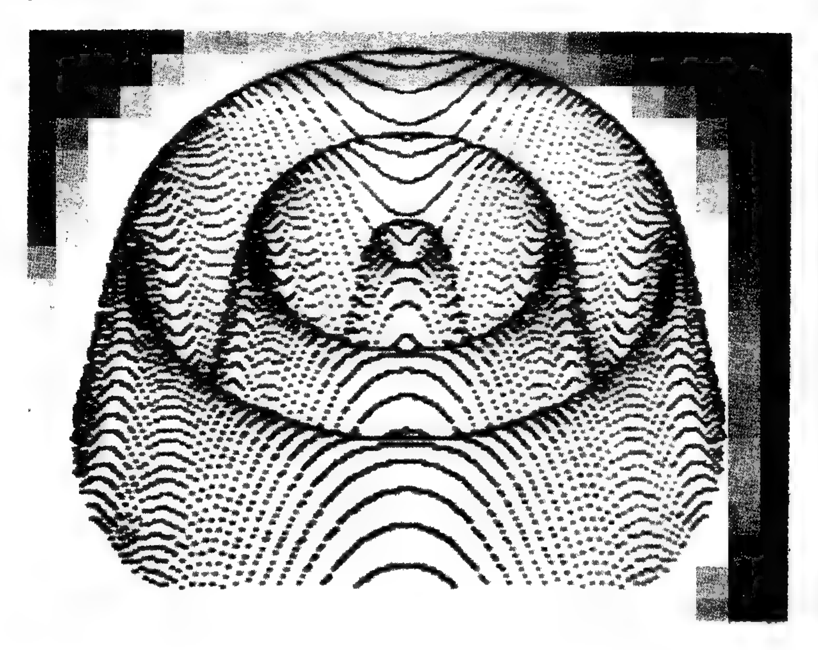
```
10 DEF FN a (z) = z*z*z*
20 PRINT FN a (37.9)
```

Este programa, calcula el cubo de 37,9 en la línea 20. También se pueden definir funciones de cadena. Por ejemplo, puede serle útil simular un comando que poseen otros ordenadores que permite obtener los últimos caracteres de una cadena. Por ejemplo:

```
10 DEF FN A$ (S$,n) = S$(N TO )
```

Veamos ahora una manera más útil de usar esta posibilidad de definir funciones. En la línea 5 hay una función que será representada por el programa. Usted puede probar de poner otras ecuaciones en la línea 5 para ver los resulta-

dos que se obtienen. En este ejemplo, existe la opción de escoger varias resoluciones (aunque sólo recomiendo entre 1 y 10). En el nivel 10 el programa tarda 10 minutos en representar la función, mientras que a nivel uno puede tardar varias horas. Este gráfico tridimensional, nos introduce a la nueva parte del libro dedicada al color, gráficos y sonido:



```
  2 REM TRES HONDONADAS
  5 DEF FN A(Z)=32*SIN (Z/6)
 10 INPUT "RESOLUCION?(1 TO 10)",R
 20 FOR X=-100 TO 100
 30 LET J=0: LET K=1
 40 LET V=R*INT (SQR ((10^4)-X*X)/R)
 50 FOR Y=V TO -V STEP -R
 60 LET Z=INT (80+FN A(SQR (X*X+Y*Y))-.
707*Y)
 70 IF Z<J THEN  GO TO 110
 80 LET J=Z
 90 PLOT X+110,Z
100 LET K=0
110 NEXT Y
120 NEXT X
130 STOP
```

La n en el primer ejemplo, y la z en este último son puramente *variables falsas*. Con esto quiero decir que es totalmente correcto definir una variable z en cualquier otra parte del programa sin que tenga nada que ver con la utilizada en DEF FN.

Segunda Parte

Gráficos, color y sonido

CAPÍTULO SEXTO

Vista y sonido

En la primera parte vimos que el Spectrum es capaz de producir ocho colores cuyos códigos van del Ø al 7:

Ø Negro
1 Azul oscuro
2 Rojo
3 Magenta
4 Verde
5 Azul claro
6 Amarillo
7 Blanco

También vimos cómo tener el borde coloreado simplemente escribiendo BORDER x, donde x es un número entre Ø y 7. También vimos que los colores pueden cambiarse globalmente (cambiando la tinta y el papel de toda la pantalla, lo cual se consigue poniendo CLS detrás de INK o de PAPER), o bien localmente (colocando los comandos dentro de una sentencia PRINT).

Para refrescar su memoria vamos a poner una frase coloreada en la pantalla:

```
1Ø PRINT INK 1; PAPER 5; AT 1Ø, 15; "ZX";
   AT 12,12; "SPECTRUM"
```

La tinta de color azul oscuro sobre papel azul claro, queda muy bien. Pero hay otras combinaciones que no. Cambie la línea anterior de modo que escriba la frase en tinta verde sobre papel azul claro. Es más difícil de leer ¿no?

Rojo y magenta tampoco combinan bien, y magenta sobre papel verde es casi imposible de leer. Por lo tanto, vaya con cuidado a la hora de escojer los colores que va a mezclar.

El ejemplo anterior nos muestra la forma más común de colorear la pantalla, pero en realidad hay dos métodos más para hacerlo. El primero de ellos, es el que yo llamo **método directo.** Este método, se basa en el hecho de que con el cursor en modo E usted puede pulsar las teclas del 0 al 7 para cambiar el papel del texto que sigue. Pulsando cualquiera de estas teclas con el cursor en modo E al mismo tiempo que se pulsa CAPS SHIFT, produce el cambio instantáneo de la tinta. Este método es un poco difícil de describir porque en él intervienen instrucciones, que no están en las teclas. Escriba este programa que no hace nada:

```
10 REM FALSO
```

Ahora use EDIT para bajar la línea, una vez abajo, coloque el cursor en modo E pulsando CAPS SHIFT y SYMBOL SHIFT simultáneamente, luego pulse una de las teclas del 0 al 7. Verá cómo el papel de debajo de las palabras REM FALSO cambia instantáneamente. Ahora cambie también la tinta pulsando una de las teclas del 0 al 7 con el cursor en modo E al mismo tiempo que pulsa CAPS SHIFT.

Lo que ha hecho en realidad es poner caracteres de control en la línea, que le dicen al Spectrum que escriba el texto en esos colores. Todo el texto que se encuentre a partir del punto en que usted introdujo los códigos habrá cambiado. Para comprobar esto, escriba otra línea de programa:

```
20 REM OTRA LÍNEA
```

Toda la línea entera, tendrá el mismo color de la tinta y del papel que la línea 10. Este modo de colorear la pantalla, se puede usar para distinguir trozos de los listados de los programas en pantalla. Para cambiar el color de la línea 20 simplemente bájela con EDIT y actúe de un modo similar.

Para volverlo como estaba tiene que hacer marcha atrás

hasta llegar al punto que se incluyeron los códigos invisibles. Por ejemplo, baje la línea 10 con EDIT. Si usted intenta mover el cursor hacia la derecha, parece que hay que pulsar cuatro veces antes de que llegue a la F de FALSO. Esto ocurre porque aunque el cursor no se mueve, está pasando por los códigos que se han introducido. Hay dos para cada color (dos por la tinta, y dos por el papel). Fácilmente puede poner más colores de tinta o de papel, aunque sólo tendrá efecto el último. Para quitar los colores, tiene que situarse a la derecha del lugar en que los puso, y pulsar DELETE (CAPS SHIFT y 0). La primera vez que pulsa DELETE aparece un interrogante a la izquierda del cursor. La segunda vez el interrogante desaparece al mismo tiempo que el color (del papel o de la tinta) que acaba de borrar. Use DELETE una vez más y aparecerá otro interrogante. Haga DELETE por última vez y el otro color desaparecerá, obteniendo así el color y la tinta originales.

Buscar estos códigos escondidos, puede ser muy difícil. Aquí hay un ejemplo de cómo uno de ellos puede hacer que un programa aparente no funcionara bien.

```
300 IF INKEY$ = "c" THEN GOTO 1000
```

Esta línea pertenece a un programa que tenía preparado para este libro. La diferencia entre lo que se supone que tiene que hacer y lo que hace realmente no es evidente en absoluto. De algún modo, se ha insertado un atributo de color entre las comillas, con la 'c', que simplemente deja el papel del mismo color en que está. El efecto es totalmente invisible, pero cuando se ejecuta el programa, no se reconoce la 'c' como el carácter que hace cumplir la condición. Una vez descubierto y borrado el atributo escondido, el programa funcionó perfectamente.

UN PEQUEÑO TRUCO: LOS LISTADOS INVISIBLES

Como habrá observado, es perfectamente posible, poner el papel y la tinta del mismo color. Por ejemplo, poner tanto la tinta como el papel de color blanco. Escriba unas pocas líneas de

programa tales como estamentos REM. Ahora haga EDIT de la primera línea, y con el cursor en modo E pulse CAPS SHIFT y 7 simultáneamente (estoy suponiendo que el color general de su papel es el blanco). ¡La línea desaparece! Además una vez pulsado ENTER, no puede ver nada del listado exceptuando el primer número de línea.

LISTando a partir de la siguiente línea, el listado se hace visible (exceptuando la primera línea), pero usted puede repetir esto fácilmente con cada línea.

Ahora vamos a escribir otra vez frases en color en la pantalla:

```
10 PRINT "Hola mundo cruel"
20 PRINT "Estoy muy contento de verte"
```

Ahora baje la línea con EDIT, y mueva el cursor hasta colocarlo dentro de las comillas, antes de la H de "Hola". Ahora, con el cursor en modo E, cambie los atributos de color y tinta, tal como hizo antes. No se preocupe porque el resto del listado también tenga los colores. Simplemente ejecute el programa con RUN. Verá como sólo la primera línea aparece con los colores nuevos. Si ahora edita la línea de nuevo y justo antes de las segundas comillas vuelve a poner el papel tal como estaba, sólo aparecerá esta frase coloreada en el listado. Esto es una ventaja sobre el modo usual de definir el color localmente, con INK y PAPER, ya que de este modo, tienen las frases en el listado tal como aparecerán después en la ejecución del programa.

El segundo método, hace uso de los caracteres de control. Si usted mira el listado de códigos ASCII al final del libro, verá que los caracteres cuyos códigos van del 6 al 23 se usan para propósitos especiales. Cuando se le dice al Spectrum que los imprima, no producen caracteres sino un efecto, normalmente cambian la posición del cursor o el color del papel o la tinta. Mirando a la tabla puede ver que todas estas funciones son posibles utilizando los caracteres de control (imprimiéndolos mediante CHR$): Coma de PRINT, cursor izquierda, cursor derecha, ENTER,

INK, PAPER, FLASH, BRIGHT, INVERSE, OVER, AT y TAB. Tanto EDIT como el cursor arriba y abajo y DELETE, no son en realidad utilizables.

La coma de PRINT, sirve para colocar los campos a imprimir en dos partes de la pantalla, la derecha y la izquierda:

```
PRINT "Hi","there","how","are","you?"
```

Esto también puede hacerse usando CHR$ 6:

```
PRINT"Hi";CHR$ 6; "there";CHR$ 6;"how";
CHR$ 6;"are";CHR$ 6 ;"you?"
```

Ya vimos cómo funcionaba CHR$ 8 al utilizar OVER. Esto hace que el cursor vuelva una posición hacia atrás, de manera que el próximo carácter se imprime encima (en inglés, OVER) del anterior. De un modo similar, CHR$ 9 mueve el cursor, una posición hacia adelante.

```
PRINT OVER 1; "O"; CHR$ 8; "/"
PRINT "G";CHR$ 9: "oof"
```

Como puede ver, CHR$ 8 es muy útil para sobreimprimir caracteres, mientras que CHR$ 9 simplemente mueve el cursor donde hubiera ido a parar de todos modos. CHR$ también es muy útil, ya que hace que el campo a imprimir lo haga en la próxima línea, es como mover el carro de la máquina de escribir.

```
PRINT "line 1";CHR$ 13;"line 2"
```

El color del papel y la tinta, también se pueden cambiar con CHR$. Aquí tiene un ejemplo (primero "normal" y luego usando los caracteres de control):

```
10 PRINT INK 2; PAPER 6; "NORMAL"
20 PRINT CHR$ 16; CHR$ 2; CHR$ 17;
   CHR$ 6; "CHR$ version"
```

Ejecute este programa y verá que las dos líneas tienen los mismos atributos de color. FLASH, INVERSE, BRIGHT

y OVER, también pueden utilizarse con CHR$ de un modo igualmente fácil:

```
10 PRINT CHR$ 18;CHR$ 1;CHR$ 16;CHR$ 2;
   CHR$ 17;CHR$ 6; "PANIC!"
```

Habrá observado que para usar este método, primero se escribe CHR$ (código) del atributo (por ejemplo CHR$ 17 para PAPER) y luego otra vez CHR$ seguido del código correspondiente. 0 a 7 para un color o bien un 1 o un 0 para la presencia o ausencia de un atributo.

Por supuesto que CHR$ 22 y CHR$ 23 hacen la función de TAB y AT respectivamente.

Usted debe estar pensando. "Todo esto está muy bien, pero ¿para qué quiero usar estos caracteres de control si parece más largo de escribir que los comandos directos como INK o PAPER?" Bien primeramente, usted no puede poner comandos como "backspace" (espacio hacia atrás). o ENTER en una línea PRINT si no es con CHR$ (al menos de un modo más fácil). Segundo: el uso de CHR$ le permite asignar a una variable de cadena todo un conjunto de textos, caracteres de control (colores, FLASH, etc.) y gráficos. Esta variable puede ser impresa en cualquier sentencia PRINT. Esto no se podría hacer sin disponer de CHR$, y es muy útil. Poner un trozo de texto, con sus colores, tabulaciones y gráficos en una cadena es una manera mucho más limpia y rápida que imprimir estas líneas en la pantalla usando las líneas normales de la sentencia PRINT. Esta diferencia de velocidad puede llegar a hacer que un juego escrito en BASIC sea aceptable o por el contrario tan lento que ya no sea interesante.

Como un ejemplo muy simple, vea cómo podría llamar a esta variable cuando fuera necesitada en el programa:

```
10 LET A$ = CHR$ 16 + CHR$ 2 + CHR$ 17 +
CHR$ 6 + "PUNTUACIÓN:"
```

Como puede ver esto produce la impresión en cualquier parte (usando PRINT AT) de la palabra "puntuación". Hubiera sido igual de fácil hacer que parpadeara (con FLASH). Observe también que cuando coloca algún CHR$ en una cadena tiene que usar '+' para separarlos en lugar del punto y coma. Note también, que la longitud de la

cadena no tiene límite, por lo tanto, es muy fácil almacenar toda una pantalla llena de texto con sus atributos de color, etc. en una variable de cadena.

Este método es sin embargo bastante lento, para agilizarlo podemos utilizar de nuevo SCREEN$, que tiene dos funciones diferentes en el Spectrum. Primero, SCREEN$ puede ser utilizado para guardar una pantalla llena de información en el cassette, y luego recuperarla de nuevo. Aquí hay un ejemplo de esto:

```
10 PAPER 3:INK 9:CLS
20 FOR N = 1 TO 64
30 PRINT " SCREEN$^";
40 NEXT N
50 SAVE "SCREEN" SCREEN$
60 CLS
70 LOAD "SCREEN" SCREEN$
```

Ejecute este programa, y una vez esté la pantalla casi llena de "SCREEN$" aparecerá el mensaje "start tape, then press any key". Compruebe que las clavijas MIC del Spectrum y del cassette están conectadas y que dispone de una cinta libre, lista para grabar. Ponga en marcha el cassette y pulse una tecla cualquiera y cuando haya terminado se borrará la pantalla (línea 60) y el programa esperará a que recupere de la cinta lo que había en la pantalla (rebobine la cinta y conecte las clavijas EAR antes de poner en marcha el cassette). Como puede ver, su pantalla se vuelve a llenar de "SCREEN$". Primero aparecen los primeros bytes de las ocho primeras líneas, luego los segundos, etc. Después ocurre lo mismo con las ocho siguientes y por último se graban los atributos de color cambiando a magenta y blanco si es que no estaban puestos. Para observar este efecto más claramente añada la siguiente línea al programa:

```
55 PAPER 7: INK 0
```

Mientras que todo esto va muy bien para guardar una pantalla de gráficos que usted haya creado, no es muy práctico, pues para utilizarlo en un programa hay que detener éste y avisar a la persona que lo usa que recupere la pantalla de la cinta. Para solucionar esto, tenemos la segun-

da función de SCREEN$. Esta función ya la utilizamos, cuando detectábamos qué carácter había en una posición determinada de la pantalla. Para guardar una pantalla llena de información (o sólo una parte) puede usar SCREEN$ (x, y) para colocar los caracteres que hay en la pantalla en una variable de cadena (un conjunto de 24 por 32) y luego simplemente imprimir la cadena en la posición 0,0. Aquí tenemos un ejemplo que almacena e imprime las cinco primeras líneas de la pantalla:

```
 10 LET X = 1
 20 FOR A = 1 TO 40
 30 PRINT "PCW rools "
 40 NEXT A
 50 DIMS$ (32*5)
 60 FOR R = 1 TO 5
 70 FOR C = 1 TO 32
 80 LET S$(X) = SCREEN$(R,C)
 85 LET X = X+1
 90 NEXT C:NEXT R
100 CLS
110 PRINT AT 0,0;S$
```

Como puede ver no es necesario calcular cuántos caracteres hay en las cinco primeras líneas de la pantalla, simplemente necesita poner 32*5. Observe que también ha creado una variable X para determinar la asignación de los caracteres a la cadena S$.

Por supuesto que este método no permite guardar de un modo fácil, los atributos de color, FLASH o BRIGHT. Para hacerlo, tiene dos posibilidades, por ejemplo puede definir una nueva variable numérica en la que se guarden los atributos de los caracteres que está almacenando y luego colocarlos (mediante POKE) directamente en el fichero de atributos (mire el mapa de memoria del apéndice para ver dónde se encuentra). Por otro lado, puede usar los caracteres de control descritos anteriormente.

Para hacer lo primero piense que necesita determinar los atributos de cada uno de los caracteres que va a almacenar. Esto puede hacerlo usando ATTR. Recuerde que ATTR, como SCREEN$, va seguido de las coordenadas de la posición en que se encuentra el carácter, encerradas entre paréntesis. ATTR devuelve un número que está com-

puesto de otros cuatro que le relacionan con FLASH, BRIGHT, INK y PAPER. El número se compone de:

128 si el cuadrado está en FLASH
64 si el cuadrado está en BRIGHT
8 multiplicado por el código del color de PAPER
el color de INK

A primera vista, la elección de estos números puede parecer un poco aleatoria. Sin embargo, esta elección, aparece mucho más clara si consideramos los números en forma binaria. Recuerde que cada columna corresponde a una potencia de 2. Es decir las columnas son:

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
qué son:	128	64	32	16	8	4	2	1

En el lenguaje de los ordenadores, estas ocho columnas se llaman **bits,** y como puede ver, el bit ocho es 128, el 7 es 64, etc. Al hablar de los atributos podemos pensar en estos ocho bits como un todo (un byte):

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
FLASH	BRIGHT	PAPER			INK		

Con esto tenemos que los bits 0, 1 y 2 indican el color de la tinta, los bits 3, 4 y 5 el color del papel y los bits 6 y 7 le dicen al Spectrum si BRIGHT y FLASH están encendidos o apagados. Veamos un ejemplo concreto para ver cómo trabaja esto:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	1
FLASH	BRIGHT	PAPER			INK		

Inmediatamente puede ver qué FLASH está encendido (el bit 7 es 1), pero que no hay brillo (BRIGHT) porque el bit 6 es 0. Los bits 5, 4 y 3 dan el número 101, que en decimal es (1*32) + (0*16) + (1*8) = 40. Si divide 40 por ocho se obtiene 5 lo cual indica que el color del papel es azul claro o cyan. Ahora el color de la tinta. El número obtenido es el 011 que actuando del mismo modo: (0*4) + (1*2) + (1*1) nos da 3, por lo tanto el color de la tinta es magenta. Y el número completo es:

(128) + (40) + (3) = 171

Para comprobarlo escriba:

PRINT INK 3; PAPER 5; FLASH 1;"anybody"

Entonces suponiendo que al menos parte de la frase se imprime en la esquina superior izquierda de la pantalla, escriba sin número de línea:

PRINT ATTR (0,0)

Esto debería dar 171.

Volviendo a nuestro problema conociendo ya como funciona ATTR, vamos a ver cómo podemos almacenar no sólo los caracteres de una parte de la pantalla sino también sus atributos.

Una solución es inmediata, si recordamos que se puede poner color en la pantalla usando los caracteres de control CHR$ 16 y CHR$ 17. Por ejemplo, este programa, almacenará tanto el texto como los atributos de las dos primeras líneas de la pantalla:

```
  10 REM CREACION DE DOS LINEAS DE CARAC
TERES
  20 PRINT  PAPER 4; INK 5;AT 0,0;"*****
***************************"
  30 PRINT  PAPER 6; INK 9;AT 1,0;",,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,"
  40 REM COLOCACION DE LOS CARACTERES Y
SUS ATRIBUTOS EN LA CADENA A$
  50 LET A$=""
```

```
 60 FOR X=0 TO 1
 70 FOR Y=0 TO 31
 80 LET B$=CHR$ 17+CHR$ INT (ATTR (X,Y)
/8)+CHR$ 16+CHR$ (ATTR (X,Y)-8*INT (ATTR
 (X,Y)/8))
 90 LET C$=CHR$ CODE SCREEN$ (X,Y)
100 LET A$=A$+B$+C$
110 NEXT Y: NEXT X
120 CLS
130 REM RE-PRINT 2 LINES
140 PRINT AT 0,0;A$
```

Por supuesto que no sería nada difícil, añadir a este programa lo necesario para detectar y almacenar los atributos de FLASH y BRIGHT, simplemente añadiendo más caracteres de control, junto con el cálculo conveniente en la línea 70.

Habrá observado, que en este programa uso INK 9. Esto coloca la tinta que mejor contrasta con el color del papel (y siempre es blanca o negra). INK 8 también es válido y lo que hace es imprimir en el color de tinta (o de papel si se pone PAPER 8) que se usó la última vez que se imprimió algo en ese cuadrado.

COLORES

Ya habrá observado, que mientras los puntos se pueden colocar (PLOT) en un nivel de resolución de 256 × 176, los colores sólo se pueden colocar por cuadrados con una resolución de 32 × 22. Un ejemplo rápido de esto último consiste en añadir un comando PAPER a una sentencia CIRCLE de este modo:

CIRCLE PAPER 4; 128,88,50

Observará que mientras la línea del círculo se dibuja en alta resolución, el color del papel, llena todo el cuadrado de un carácter. Sólo se puede tener un color de tinta y uno de papel en cada cuadrado, lo que significa que el Spectrum no permite la alta resolución en color.

Una vez dicho esto, de que sólo se puede tener un color

de papel y de tinta por cuadrado, diremos también que el Spectrum tiene una peculiaridad que parece que pueda haber dos colores de tinta en un mismo cuadrado:

```
 10 REM DOS TINTAS POR CARACTER
 15 REM
 20 BORDER 1
 30 FOR x=0 TO 7
 40 FOR y=0 TO 7
 50 IF y=x THEN  NEXT y
 55 PAUSE 100
 60 INK y: PAPER x: CLS
 70 PLOT 0,100: DRAW 255,0
 80 PLOT 0,98: DRAW 255,0
 90 PRINT  OVER 1;AT 9,5;"██"
100 PLOT 0,70: DRAW 255,0
110 PLOT 0,67: DRAW 255,0
120 PRINT  OVER 1;AT 13,10;"██"
130 NEXT y: NEXT x
```

Como ve esta peculiaridad parece contradecir el ejemplo anterior, ya que pueden aparecer líneas de distinto color en el mismo cuadrado dependiendo únicamente de la distancia en puntos que haya de la una a la otra. Por cierto a los puntos de la resolución, se les conoce con el nombre de "pixels".

128 COLORES

Hemos visto que el Spectrum puede producir 8 colores distintos, es decir, seis colores, más el blanco y el negro. ¿Pero, qué hay sobre 64 colores?, ¿y sobre 128? Pues bien, es posible generar muchos más colores mezclando los ocho ya existentes a nivel de pixels. Para hacer esto, necesita crear un gráfico definido por usted, que parece un pequeño tablero de ajedrez. Para ello vea el programa de creación de gráficos que se muestra más tarde. Haciendo el papel y la tinta de dos colores distintos aparecen mezclados, ya que los puntos están muy juntos. Este programa, le muestra toda la gama de colores que se pueden obtener haciendo esto. El número puede ser doblado si añadimos el factor brillo, creando versiones más apagadas o brillantes de cada color. Tenga en cuenta que la A de

las líneas 70 y 120 es un carácter gráfico de los definidos por el usuario.

```
  2 REM   128 colores
  5 REM juegos de colores
 10 FOR a=0 TO 6 STEP 2
 20 POKE USR "a"+a,BIN 01010101
 25 POKE USR "a"+a+1,BIN 10101010
 30 NEXT a
 40 FOR p=0 TO 7
 50 FOR i=0 TO 7
 60 FOR b=0 TO 1
 70 PRINT  PAPER p; INK i; BRIGHT b;"aa
";p;i;: NEXT b
 80 NEXT i: NEXT p
 90 REM combinaciones de color
100 FOR p=0 TO 7: FOR i=0 TO 7: FOR b=0
TO 1
110 FOR y=0 TO 31
120 PRINT  PAPER p; INK i; BRIGHT b;"aa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
130 NEXT y
140 POKE 23692,255
150 NEXT b: NEXT i: NEXT p
```

EL FICHERO DE LOS ATRIBUTOS

Hay un modo más de cambiar los colores en el Spectrum que consiste en colocarlos directamente (mediante POKE) en un área de la memoria llamada fichero de atributos (atribute file). Este fichero, se encuentra justo detrás de la memoria de pantalla en las posiciones que van desde la 22528 hasta la 23296. Un poco de matemáticas nos revelan que esta área de la memoria consta de 768 posiciones que es por supuesto 32 × 24. En otras palabras, hay una posición de memoria en el fichero de atributos, para cada posición de la pantalla. Normalmente, sólo se usan las 22 líneas superiores, quedando las dos inferiores para mensajes de error, INPUT, etc.

Pero "colocar directamente en el fichero de atributos" ¿qué significa exactamente? Pues bien, **POKE** describe el proceso de poner un trozo de información (llamado un byte, ¿recuerda?) en una posición de memoria especificada. Su opuesto, es **PEEK**, ya que cuando al Spectrum le

damos la instrucción PRINT PEEK (número de posición) nos imprime el valor del byte que se encuentra en esa posición de memoria. Esto es lo que ocurre realmente cuando usamos ATTR para obtener los atributos de un carácter en la pantalla. Para clarificar esto un poco más, comprobaremos que ambas cosas son lo mismo:

```
10 PRINT INK2;PAPER6;AT0,0;"*"
20 PRINT ATTR(0,0)
30 PRINT PEEK 22528
```

Ejecute esto, y verá que además de obtener una estrella roja sobre fondo amarillo, tanto la línea de ATTR como la de PRINT PEEK devuelven el valor de 50. Ahora para ver cómo se puede usar el fichero de atributos, escriba lo siguiente:

```
POKE 22528,42
```

Mágicamente (e instantáneamente) la estrella está en rojo sobre fondo azul claro. Intente hacer lo mismo pero con otros números (hasta 255) para ver el efecto que esto produce sobre el recuadro en que se encuentra el asterisco (que como habrá deducido, corresponde a la posición 22528 de la memoria, claro dentro del fichero de atributos). Observe que ponga lo que ponga en esta posición, el asterisco no desaparece, aunque lo puede parecer si usted hace que el color del papel sea el mismo que el de la tinta.

Un uso inmediato de lo que acabamos de ver, es cambiar los colores del papel y la tinta (ya sea en toda la pantalla o sólo una parte), sin alterar para nada el texto ni los gráficos. Recordará que cuando esto se efectúa mediante un comando, ya sea local o global, se destruye el texto o el gráfico que había en la posición.

Aquí hay una rutina muy simple para cambiar el color y la tinta a los dos colores escogidos. Aunque lo hace para toda la pantalla, es muy fácil cambiarla para que sólo lo haga sobre una parte.

```
10 REM CAMBIO DE ATRIBUTO
20 FOR A = 22528 TO 23295
30 POKE A, N
40 NEXT A
```

(Observe que N es el número de ATTR. Por ejemplo, N = 47 produce papel azul claro y tinta blanca).

```
10 REM CAMBIO DE PAPEL
20 DIM A$ (704)
30 PRINT AT 10,0; INK 7; PAPER 7; "¡Sorpresa!"
40 PAUSE 50
50 PRINT OVER 1; AT 0,0; A$
```

Aparte del método que usa POKE, existe otra manera de cambiar los atributos que consiste en usar PRINT OVER, imprimiendo una cadena de espacios del tamaño de la pantalla para cambiar el papel. Observe que para cambiar la tinta tendrá que hacer PRINT con una cadena de cuadrados (los de la tecla 8 en modo gráfico). Este ejemplo nos muestra el uso de PRINT OVER:
En la línea 80, también se podría incluir BRIGHT 1, FLASH 1 o INVERSE 1. Este último método, es mucho más rápido si usted sólo desea cambiar el papel y una vez se ha creado A$. Más adelante en este libro, veremos una rutina en código máquina para cambiar el color del papel y la tinta casi instantáneamente sin destruir el texto.

DIBUJOS EN COLOR

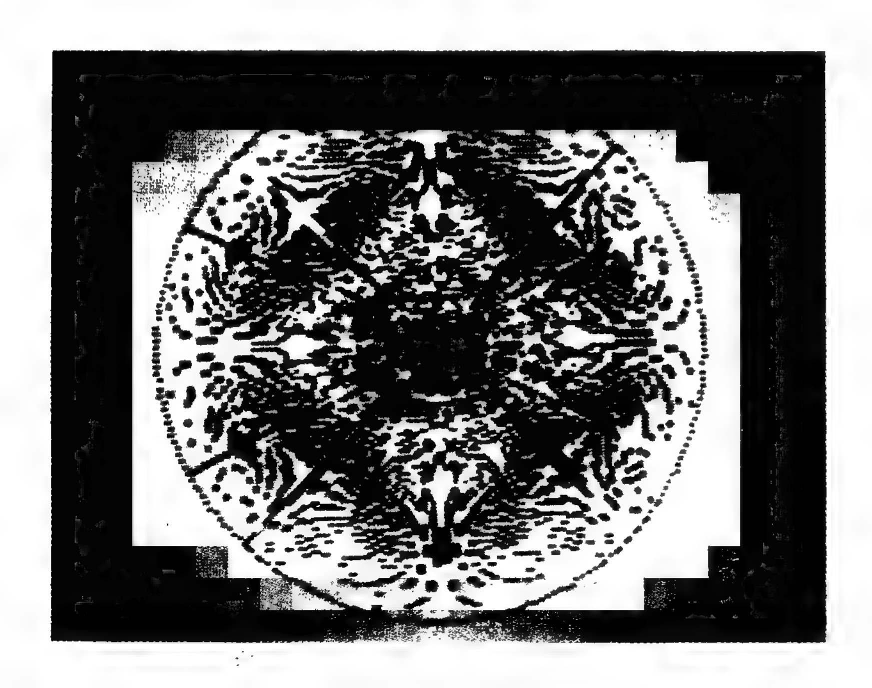
Vamos a dibujar algo en alta resolución en color. ¿Qué hay más indicado que un arco iris?

```
  5 REM  ARCO-IRIS
 10 LET X=1
 20 DIM A(6): GO SUB 120
 30 LET K=1
 40 PAPER 7: BORDER 0: CLS
 50 FOR Y=160 TO 250 STEP 15
 60 FOR J=1 TO 5
 70 INK A(K): PLOT Y+J,0: DRAW -Y,Y-80+
J,PI/2
 80 NEXT J
 90 LET K=K+1
100 IF K=7 THEN  GO TO 200
110 NEXT Y
120 LET A(1)=3: LET A(2)=1: LET A(3)=5:
LET A(4)=4: LET A(5)=6: LET A(6)=2
```

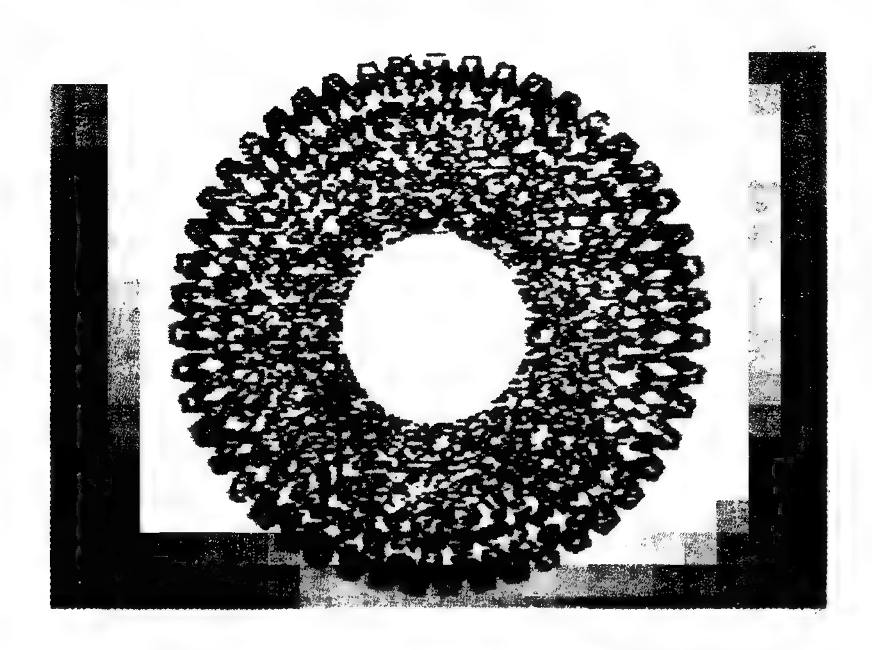
```
130 RETURN
200 PRINT  PAPER 7;AT 19,5; INK 3;"S";
INK 1;"P"; INK 5;"E"; INK 4;"CT"; INK 6;
"R"; INK 2;"UM"
```

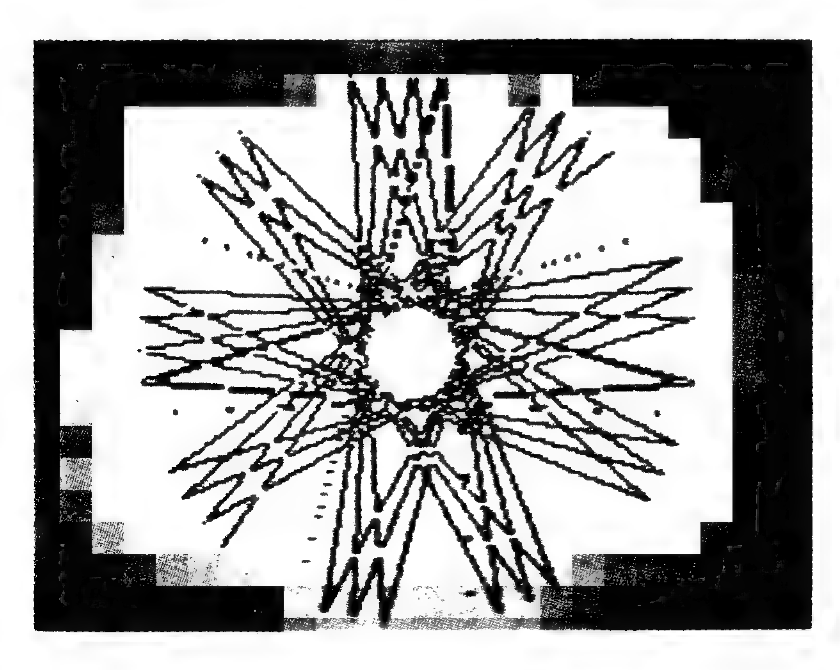
Observe cómo el hecho de que DRAW dibuja arcos, se utiliza aquí para dibujar las curvas del arco iris. Observe también que las curvas están a distancia de un cuadrado para evitar el tener más de una tinta en cada cuadrado. Dibujar atractivos cuadros en color es sorprendentemente fácil con el Spectrum. El siguiente programa sólo tiene una línea de longitud, pero usted se sorprenderá de lo que hace. Cambie el papel a amarillo (6) y la tinta a azul oscuro (1). Esta rutina tan simple fue desarrollada por Andrew Glaister, con algunas modificaciones mías:

```
1 PLOT 65,27: DRAW OVER 1; 120,120,59↑3*PI
```

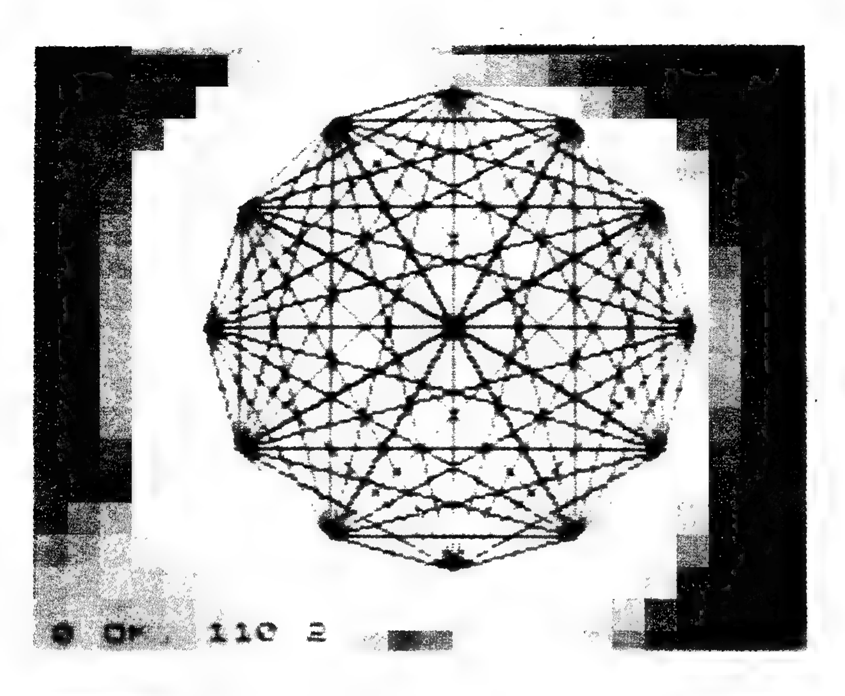


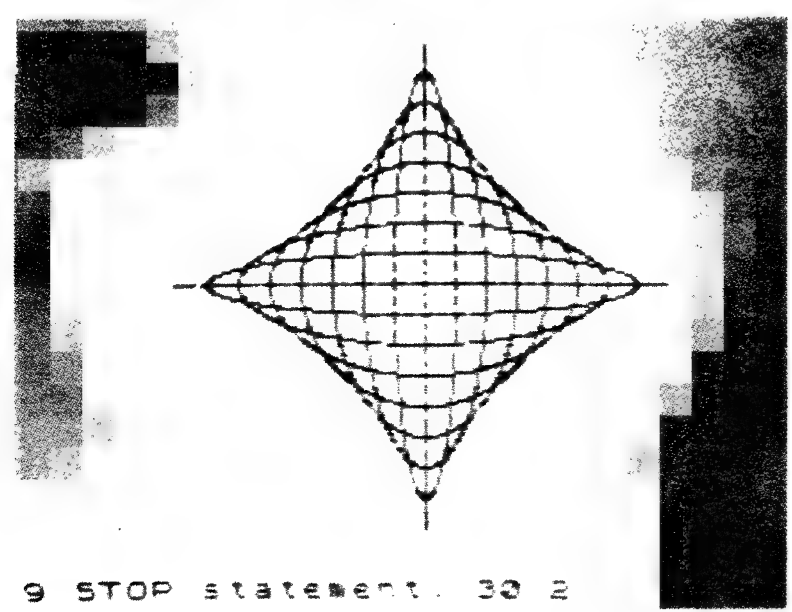
Pruebe a variar el número 59 de la línea y juegue con las potencias 2 y 3. Aquí hay dos ejemplos más que demuestran que este programa es muy versátil:

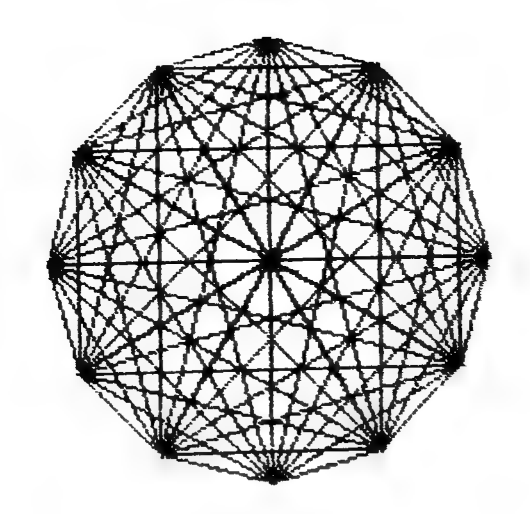




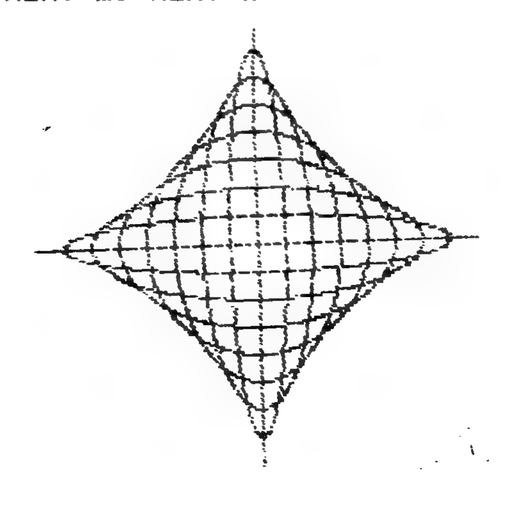
Otros dibujos extremadamente atractivos se pueden generar con sólo unas pocas líneas de BASIC. Aquí tenemos algunos ejemplos hechos por G. Scott:



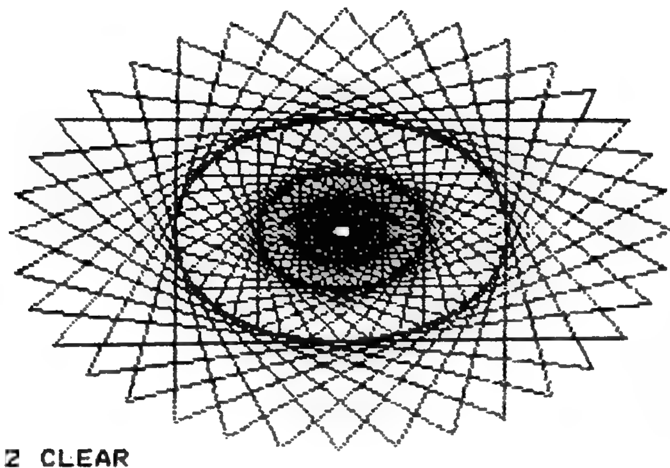




```
  2 CLEAR
  5 DIM a(12): DIM b(12)
 10 FOR n=1 TO 12
 20 LET k=n/6*PI
 30 LET a(n)=128+80*SIN k: LET b(n)=86+
80*COS k
 40 PLOT a(n),b(n)
 50 NEXT n
 60 FOR n=1 TO 12
 70 FOR m=1 TO 12
 80 LET ox=a(m)-a(n)
 90 LET oy=b(m)-b(n)
100 PLOT a(n),b(n): DRAW ox,oy
110 NEXT m: NEXT n
```



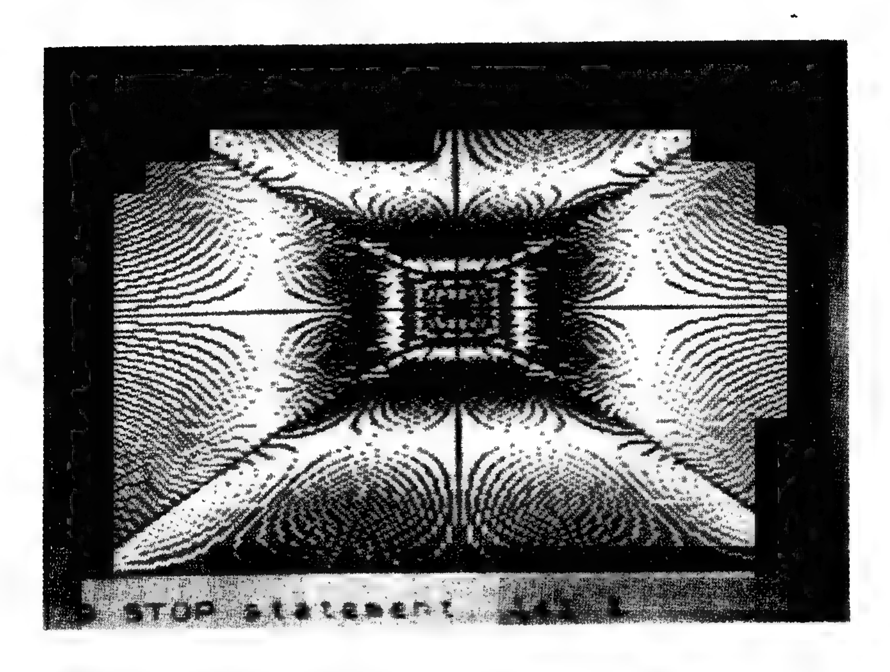
```
2 CLEAR
3 INK 2
5 LET x=0: LET y=80
10 FOR n=0 TO 2*PI STEP PI/180
15 PLOT 128+x*SIN n,87+y*COS n
20 NEXT n
25 LET x=x+10: LET y=y-10
30 IF y=-10 THEN  STOP
40 GO TO 10
```



```
  2 CLEAR
 10 DIM a(36): DIM b(36)
 15 LET l=120: LET j=80
 17 FOR h=1 TO 5
 20 FOR n=1 TO 36
 30 LET k=n/18*PI
 40 LET a(n)=128+l*SIN k: LET b(n)=88+j
*COS k
 45 PLOT a(n),b(n)
 50 NEXT n
 60 FOR n=1 TO 36
 65 LET m=n+12
 70 IF m>36 THEN  LET m=m-36
 80 LET ox=a(m)-a(n)
 90 LET oy=b(m)-b(n)
100 PLOT a(n),b(n): DRAW ox,oy
110 NEXT n
120 LET l=l/2: LET j=j/2
125 NEXT h
```

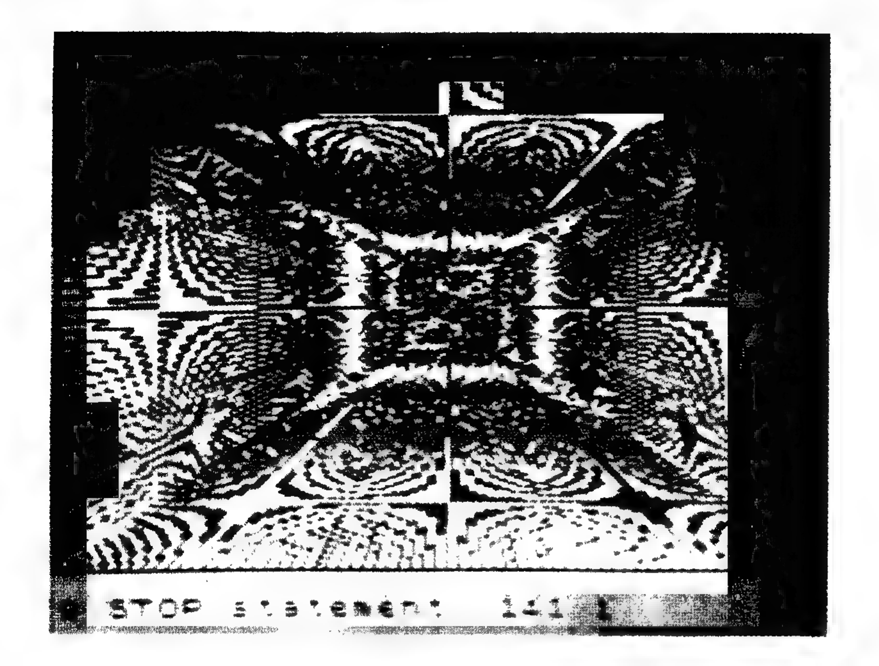
También es muy fácil llenar la pantalla de impresionantes dibujos, simplemente explotando la manera que tiene el

Spectrum de dibujar líneas. Aquí hay dos rutinas que producen resultados muy similares por métodos distintos:




```
  2 REM INTERFERENCIAS
 10 OVER 1
 20 LET z=1
 30 LET i=(RND*5)+1
 40 LET p=(RND*5)+1
 50 IF INT i=INT p THEN  GO TO 30
 60 PAPER p: INK i: CLS
 65 FOR a=1 TO 2
 70 FOR x=0 TO 254 STEP 2
 80 PLOT 128,88: DRAW (-127*z)+(x*z),z*
-87
 90 NEXT x
100 FOR y=0 TO 175 STEP 2
110 PLOT 128,88: DRAW 127*z,z*-87+(y*z)
120 NEXT y
130 LET z=-z
140 NEXT a: PAUSE 100: GO TO 20
```

Añadiendo OVER 1 con STEP 0.8 produce resultados aún más impresionantes:



Y una manera muy simple de obtener un resultado similar:

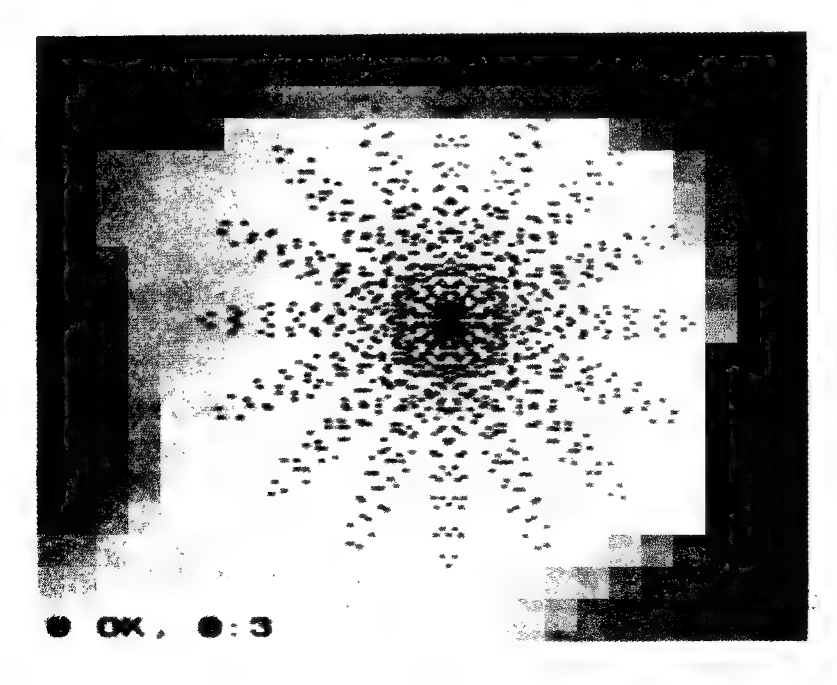
```
 5 REM  TUNEL DEL TIEMPO
10 FOR A=0 TO 255
20 PLOT A,0
30 DRAW  OVER 1;2*(127.5-A),175
40 PLOT 0,A*175/255
50 DRAW  OVER 1;255,2*(87.5-A*175/255)
60 NEXT A
```

Lo que es bastante interesante sobre estos dibujos es que la "nieve" de la pantalla puede producir un efecto muy agradable si la combinación de colores es buena.

En el próximo programa, se dibuja un copo de nieve con brazos radiales que parten del centro. Pruebe a alterar algunas líneas para cambiar el número de brazos del copo de nieve, etc.

```
10 REM Copo de nieve
20 INK 7: BORDER 3: PAPER 1: CLS
30 LET p$=" ": LET t$=" "
40 FOR a=0 TO 702
```

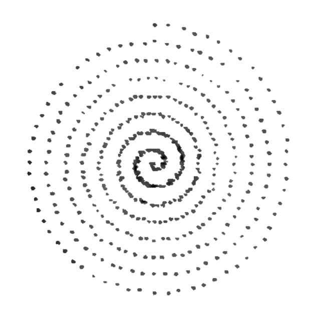
```
 50 LET p$=p$+t$
 60 PRINT ".";: NEXT a
 70 REM Crea una cadena de espacios del
tamanno de la pantalla
 80 CLS
 90 LET flag=0
100 FOR p=20 TO 80 STEP 1
110 LET y=0: LET z=PI
120 LET c=100
130 LET e=(z-y)/c
140 FOR q=y TO z STEP e
150 LET s=p*COS (q*6)
160 LET h=s*SIN q
170 LET v=s*COS q
180 PLOT 128+h,88+v
190 NEXT q
200 LET flag=flag+1
210 IF flag=101 THEN  GO TO 230
220 GO TO 150
230 LET flag=0: NEXT p
240 LET col=INT (RND*6)+1
250 PRINT AT 0,0; PAPER col; OVER 1;p$
260 PAUSE 100: GO TO 240
```



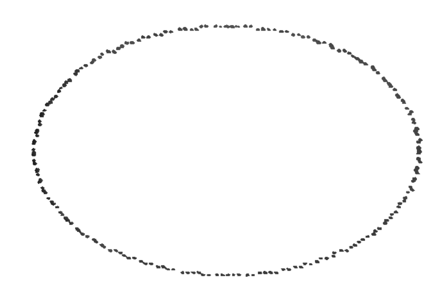
Simplemente dibujar círculos de diferentes colores puede producir un resultado muy bonito:

```
 2 REM circulos
 5 PAPER 0: CLS
10 LET x=(234*RND)+10
20 LET y=(154*RND)+10
30 LET z=(5*RND)+1
40 CIRCLE  INK z;x,y,10
50 GO TO 10
```

También se pueden dibujar fácilmente elipses y espirales:



```
  5 REM  espiral
 10 BORDER 1: PAPER 0: INK 7: CLS
 20 LET r=50
 30 FOR f=0 TO 500
 40 LET a=(RND*7)+1
 50 LET r=r-0.1
 60 PLOT  INK a;128+r*SIN (f/32*PI),88+
r*COS (f/32*PI)
 70 NEXT f
```



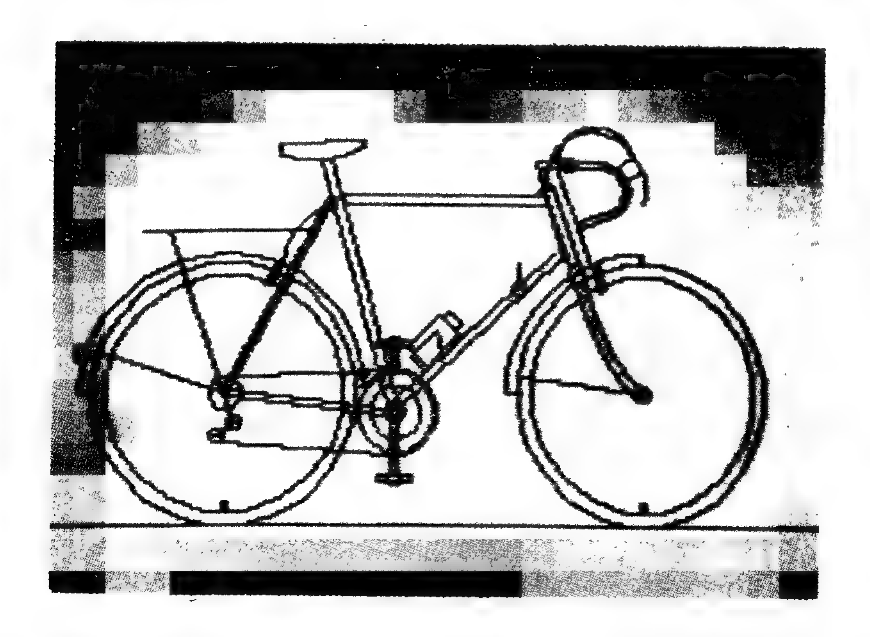
```
 10 REM DIBUJA UN OVALO
 20 REM LOS RADIOS DEBEN SER MENORES A
LOS LIMITES DESDE EL CENTRO
 30 REM LOS LIMITES DEL CENTRO SON 128,88
```

```
40 BORDER 1: PAPER 5: INK 1: CLS
50 INPUT "DOS RADIOS ?";R1;R2
60 FOR N=0 TO 200
70 PLOT 128+R1*SIN (N/100*PI),88+R2*CO
S (N/100*PI)
80 NEXT N
```

También es posible hacer dibujos animados en alta resolución:

```
10 REM  pendulo
20 BORDER 1: PAPER 1: INK 7: CLS
30 FOR x=1 TO 3000 STEP 0.5
40 PLOT 128,160: DRAW -120*SIN (x/10),
110,2*PI
50 NEXT x
```

Para demostrar que el Spectrum también puede llegar muy lejos en cuanto a dibujo realista se refiere, aquí tiene una escelente imagen de una bicicleta de carreras hecha por G. Scott y R. Turner.



```
2 REM *****La bicicleta*****
3 REM  G.J.Scott&R.I.Turner
4 REM *****La bicicleta*****
```

```
  10 CIRCLE 60,70,40: CIRCLE 60,70,5
  15 PLOT 103,70: DRAW -86,0,PI: DRAW 3,
0
  16 PLOT 60,70: DRAW -38,13
  20 CIRCLE 190,70,40
  25 PLOT 150,70: DRAW -3,0: DRAW 43,43,
-PI/2: DRAW 0,-3
  26 PLOT 190,70: DRAW -38,5
  30 CIRCLE 60,70,36: CIRCLE 190,70,36
  35 PLOT 0,30: DRAW 255,0
  40 CIRCLE 112,64,12: CIRCLE 190,70,2
  45 PLOT 18,80: DRAW -2,0: DRAW 0,5: DR
AW 3,0
  50 PLOT 60,69: DRAW 50,-6
  55 PLOT 60,71: DRAW 50,-6
  70 CIRCLE 112,64,3: CIRCLE 112,64,8
  80 PLOT 59,70: DRAW 32,60
  90 PLOT 61,70: DRAW 32,60
 100 DRAW -1,0: DRAW 19,-67
 110 DRAW 3,1: DRAW -19,67
 120 DRAW 64,0
 130 PLOT 97,128: DRAW 63,0
 140 DRAW -1,5: DRAW 8,-24
 150 DRAW 3,1: DRAW -8,24
 160 DRAW -3,-1: DRAW 6,-21
 170 DRAW -52,-47: DRAW -3,1
 180 DRAW 52,47
 190 CIRCLE 63,61,2: CIRCLE 58,57,2
 200 PLOT 60,70: DRAW 3,-9: DRAW -5,-4
 210 PLOT 58,55: DRAW 53,-4
 220 PLOT 112,51: DRAW 0,26,PI
 230 DRAW -47,-2
 240 CIRCLE 151,102,2: PLOT 151,102: DRA
W 0,8
 250 CIRCLE 170,105,2: PLOT 169,103: DRA
W 4,-12: DRAW 16,-22,PI/5
 260 PLOT 172,103: DRAW 4,-12: DRAW 15,-
20,PI/5
 270 PLOT 168,104: DRAW -1,4: PLOT 172,1
04: DRAW -1,5
 280 PLOT 102,72: DRAW 7,4,-PI/6
 290 PLOT 102,74: DRAW 7,4,-PI/5
 300 PLOT 111,63: DRAW 0,-17
 310 PLOT 113,63: DRAW 0,-17
```

```
 320 CIRCLE 112,44,2
 330 PLOT 107,45: DRAW 10,0
 340 DRAW 0,-2: DRAW -10,0: DRAW 0,2
 350 PLOT 111,78: DRAW 0,8
 360 PLOT 113,78: DRAW 0,8
 370 CIRCLE 112,85,2: PLOT 107,84: DRAW
10,0: DRAW 0,2: DRAW -10,0: DRAW 0,-2
 380 PLOT 159,134: DRAW -2,8
 390 DRAW 21,0: DRAW 0,-2,-PI: DRAW -19,
0: DRAW 2,-6
 400 PLOT 167,141: CIRCLE 167,141,2
 410 PLOT 178,142: DRAW 0,-18,-PI
 420 DRAW -8,-3: DRAW 0,2: DRAW 8,3: DRA
W 0,14,PI
 430 PLOT 184,141: DRAW 4,3: DRAW 3,-4:
DRAW -4,-3
 440 PLOT 191,140: DRAW 0,-11,-PI/4
 450 PLOT 188,144: DRAW -28,-7,PI/1.2: D
RAW -8,-6,-PI/2
 455 PLOT 188,144: DRAW -24,-6,PI: DRAW
13,-37: DRAW 1,1: DRAW -3,8: DRAW 1,0: D
RAW 2,-7
 460 PLOT 92,130: DRAW -3,12
 470 DRAW 3,0: DRAW 3,-12: DRAW -3,0
 480 PLOT 76,143: DRAW 26,3,PI/4: DRAW 0
,2,-PI: DRAW -26,-1: DRAW 0,-4
 490 PLOT 95,130: DRAW -18,-20,PI/3
 500 DRAW -3,-7: DRAW -1,0: DRAW 3,7: DR
AW -1,0: DRAW -3,-7
 510 PLOT 85,120: DRAW -50,0: PLOT 60,70
: DRAW -17,50
 520 PLOT 122,78: DRAW -6,6: DRAW 10,10:
 DRAW 6,-6: DRAW -8,-8: DRAW 8,8: DRAW -
1,1: DRAW 2,2: DRAW -4,4: DRAW -1,-1
 530 PLOT 121,77: DRAW -4,4: DRAW 8,8: D
RAW 2,-7
 540 PLOT 60,34: DRAW 0,3: DRAW 1,0: DRA
W 0,-3
 550 PLOT 190,34: DRAW 0,3: DRAW 1,0: DR
AW 0,-3
```

Para finalizar esta colección de dibujos en alta resolución, un par de cosas que aunque el programa es muy corto, tardan un poco en ejecutarse:

```
 10 REM el sombrero de Merlin
 20 BORDER 2: PAPER 6: INK 2: CLS
 30 LET x=128: LET y=88
 40 LET m=0
 50 LET r1=70: LET r2=20
 60 FOR f=m TO 200-m
 70 PLOT x+r1*SIN (f/100*PI),y+r2*COS (
f/100*PI)
 80 NEXT f
 90 LET y=y-2
100 LET r1=r1-1: LET r2=r2-1
110 LET m=50: GO TO 60
```

```
 10 REM la isla magica
 20 REM necesita varias horas
 para ejecutarse
 30 LET x=126: LET y=40
 50 BORDER 1: PAPER 0: INK 7: CLS
 60 LET r1=70: LET r2=20
 70 FOR f=50 TO 150
 80 PLOT  OVER 1;x+r1*SIN (f/100*PI),y+
r2*COS (f/100*PI)
 90 NEXT f
100 LET r1=r1-1: LET r2=r2-1
120 GO TO 70
```

LOS LÍMITES DE DRAW

En los dibujos en alta resolución es necesario controlar que los puntos colocados en la pantalla no rebasen el área de PAPER. Es decir, que las coordenadas horizontales estén entre Ø y 255 y las verticales entre Ø y 175.

Dado que la línea que dibuja DRAW parte de un punto determinado (el último punto PLOTeado) y termina en otro también determinado (en la propia instrucción) usted debe asegurarse de que la coordenada horizontal del PLOT inicial más la primera coordenada de DRAW no exceda de 255. De un modo similar, las verticales, no deben sumar más de 175. Habiendo puesto el primer punto en el centro de la pantalla, sólo puede dibujar hasta los puntos ±127 horizontalmente y ±87 verticalmente. Estas reglas son la base de la mayoría de gráficos que imprimen puntos en posiciones aleatorias.

En algunos de estos gráficos en alta resolución los mejores colores consisten en tinta clara sobre fondo oscuro. Sin embargo, cuando se usa COPY para imprimir el dibujo en papel, nos encontramos con que éste pierde detalle y parece todo negro. El remedio es cambiar la tinta y el papel en toda la pantalla. Esto se puede hacer intercambiándolos mediante INVERSE, ya sea en cada línea o globalmente. También se puede intentar hacer uso de POKE para colocar los colores directamente en el fichero de atributos.

Para cambiar la tinta y el papel, puede usar la siguiente rutina (veremos más adelante que se puede hacer lo mismo en código máquina y que va mucho más rápido):

```
10 FOR A = 16384 TO 22528
20 POKE A,255 - PEEK A
30 NEXT A
```

Con todo lo que hemos visto sobre dibujo, se puede observar que el Spectrum tiene la suficiente potencia para hacer complicados diseños gráficos. Por ejemplo, considere este programa que crea triángulos sólidos de dimensiones dadas:

```
  1 REM TRIANGULO SOLIDO
  5 BORDER 1: PAPER 7: CLS
 10 INPUT "COORDENADA"; X
 20 INPUT "COORDENADA"; Y
 30 INPUT "¿ALTURA?"; H
 35 IF Y + H > 175 THEN GO TO 100
 40 INPUT "¿LONGITUD?"; L
 45 IF × +L < 255 THEN GO TO 100
 50 INPUT "¿COLOR?"; C
 55 CLS
 60 FOR P = 0 TO H
 70 PLOT X, Y: DRAW INK C; L, P
 80 NEXT P
 90 STOP
100 CLS: PRINT "DEMASIADO" " "ENTER AGAIN"
110 GO TO 10
```

Esta rutina que ahora sigue, le permitirá dibujar lo que quiera en la pantalla de un modo similar a un "telesketch", y luego colorearlo. Observe que dado que los cuadrados de los caracteres determinan los distintos colores puede haber partes de su dibujo en que los colores se

salgan de la línea que deseamos que los envuelva (esto ocurre cuando la línea no coincide con un cuadrado). Por lo tanto es aconsejable hacer coincidir en lo posible, el dibujo con los cuadrados. Para ayudarle a hacer esto añada al siguiente programa otra rutina que convierta la pantalla en una parrilla que marque las líneas izquierdas de cada cuadrado verticalmente y las de abajo horizontalmente. Para ello sólo necesitará crear un gráfico, ya que el segundo ya existe (use SYMBOL SHIFT y 0). El gráfico que necesita crear deberá tener todos los bytes iguales a 128, lo que lo hace muy fácil de crear (para ello puede ver el próximo apartado).

```
    2 BORDER 1: PAPER 7: INK 0: CLS : LET
  q=0
    3 PRINT AT 8,0;"pulsa 'S' para pintar
 ": PRINT AT 1,1;"Dibuja una figura usand
 o         las flechas";AT 4,0;" pulsa 'O
 ' para borrar";AT 5,0 ;"'W' para volver
 a entrar en modo dibujo"
    4 PRINT ''"pulsa una tecla para empez
 ar": PAUSE 0: CLS : LET x=128: LET y=88
    5 DIM a$(32): PLOT  INK q;x,y
    6 IF INKEY$="5" THEN  LET x=x-1: PAUS
 E 25
    7 IF INKEY$="6" THEN  LET y=y-1: PAUS
 E 25
    8 IF INKEY$="7" THEN  LET y=y+1: PAUS
 E 25
    9 IF INKEY$="8" THEN  LET x=x+1: PAUS
 E 25
   10 IF INKEY$="s" THEN  GO TO 21
   11 IF INKEY$="0" THEN  LET q=6
   12 IF INKEY$="w" THEN  LET q=0
   13 GO TO 5
   15 PLOT 100,50: DRAW 50,50: DRAW -50,3
 0: DRAW 0,-80
   20 PRINT AT 0,0;"

                                  "
   30 INPUT "x coord?";x: INPUT "y coord?
 ";y: PLOT x,y: INPUT "OK?";q$: IF q$="n"
  THEN  GO TO 900
   35 GO TO 2000
```

```
 36 REM ****PINTANDO****
 40 INPUT "que tinta?";i
 50 PRINT AT 0,0;"Pintando..."
 60 LET g=y
 90 PRINT AT 1,0;a$
100 LET z=x
105 PLOT  INK i;z,y
110 LET z=z+1
115 IF POINT (z,y)=1 THEN  GO TO 200
120 IF POINT (z,y)=0 THEN  PLOT  INK i;
z,y
130 GO TO 105
200 LET z=x
210 LET z=z-1
220 IF POINT (z,y)=1 THEN  GO TO 300
230 IF POINT (z,y)=0 THEN  PLOT  INK i;
z,y
240 GO TO 210
300 LET y=y+1
310 IF POINT (x,y)=1 THEN  GO TO 400
320 GO TO 100
400 LET y=g
405 LET z=x
410 LET y=y-1
420 PLOT  INK i;x,y: LET x=x+1
430 IF POINT (x,y)=1 THEN  GO TO 500
440 IF POINT (x,y)=0 THEN  PLOT  INK i;X,Y
450 GO TO 420
500 LET x=z
510 LET x=x-1
520 IF POINT (x,y)=1 THEN  GO TO 600
530 IF POINT (x,y)=0 THEN  PLOT  INK i;
x,y
540 GO TO 510
600 LET x=z
610 LET y=y-1
620 IF POINT (x,y)=1 THEN  GO TO 700
630 GO TO 420
700 PRINT AT 0,0;"


710 PRINT AT 0,0;"FIN"
800 INPUT "Otro dibujo ?";q$
810 IF q$="y" THEN  GO TO 25
820 STOP
```

```
 900 PLOT  OVER 1;x,y: GO TO 30
1997 REM ****POSICIONARSE****
2000 PRINT AT 0,0;"MUEVE 5,6,7 O 8 PARA
POSICIONARSE 's' PARA SALIR.
  'p' para pintar"
2100 PLOT x,y: PAUSE 10: PLOT  OVER 1;x,
y
2200 IF INKEY$="5" THEN  LET x=x-1
2300 IF INKEY$="8" THEN  LET x=x+1
2400 IF INKEY$="6" THEN  LET y=y-1
2500 IF INKEY$="7" THEN  LET y=y+1
2600 IF INKEY$="s" THEN  GO TO 30
2650 IF INKEY$="p" THEN  GO TO 3000
2700 GO TO 2100
3000 PRINT AT 0,0;"
          "

3010 GO TO 40
```

CONSTRUCCIÓN DE CARACTERES GRÁFICOS

Lo que aquí sigue, es un programa que se hace indispensable para todos aquellos que deseen crear sus propios caracteres gráficos (marcianitos, figuras técnicas, alfabeto griego, etc.). En la pantalla aparece un tablero de 8 × 8 y los posibles caracteres definidos por el usuario debajo de la letra a la que corresponden en modo gráfico. Al lado, a la izquierda de cada hilera se encuentra el valor del byte correspondiente. A la derecha hay un resumen de los comandos que usted puede utilizar en este programa. Usted puede mover el cursor (asterisco) a través del tablero usando las flechas (teclas 5 a 8) sin SHIFT. El asterisco se moverá en la dirección de la flecha que hay sobre cada tecla. En cualquier momento puede cambiar el estado del recuadro en que se encuentra el asterisco pulsando la tecla C. Si el recuadro está vacío, se llenará, y si está lleno se volverá blanco. Al mismo tiempo que hace esto, los bytes de cada hilera bajo la columna DATA cambiarán según las modificaciones que haga.

Una vez ha obtenido el caracter que desea, puede conservarlo pulsando la tecla K. Entonces el Spectrum le preguntará en qué tecla lo quiere (de la A a la U). Pulsando

ENTER el carácter reemplazará instantáneamente a la letra elegida en la lista que hay en la parte alta de la pantalla. Aquí es donde usted comprobará en pequeño, si queda bien lo que ha construido. Puede insertar en el tablero cualquiera de los gráficos definidos pulsando la tecla I. Esto necesitará hacerlo tanto si desea modificarlo como imprimirlo.

Para tener una copia del carácter en la impresora simplemente pulse P. Esto coloca una copia del carácter en la impresora así como los bytes que forman cada línea, tanto en decimal como en binario. El listado en binario es útil porque permite ver la apariencia del carácter ocho veces mayor.

Finalmente, es posible también grabar en cassette los caracteres construidos, así como recuperarlos más tarde. Esto se hace saliendo del programa (pulsando la tecla Q) y utilizando el comando SAVE "xxx" CODE. Las direcciones de memoria que usted quiere grabar tienen que ir detrás de la palabra CODE, y usted ya sabe que son USR "a" para el gráfico de la tecla A, USR "b" para el de la B, etc. Luego hay que poner el número de posiciones de memoria (o para ser exactos, el número de bytes) que va a grabar. Son siempre ocho por cada carácter. Por lo tanto, para grabar todo el juego tendrá que escribir:

```
SAVE "caracteres" CODE USR "a", (21*8)
```

Esto es, mientras que el primer número después de CODE nos dice la primera posición que se graba, el número de después de la coma indica el número de bytes que quiere grabar. En este caso, usted sabe que son 21 caracteres de 8 bytes cada uno. Observe que no hay que hacer el cálculo, pues el Spectrum lo hace por usted directamente.

Un carácter dado puede ser grabado con un comando del tipo:

```
SAVE "carácter A" CODE USR "a",8
```

(usted escoge el nombre del programa mientras tenga menos de 10 caracteres)

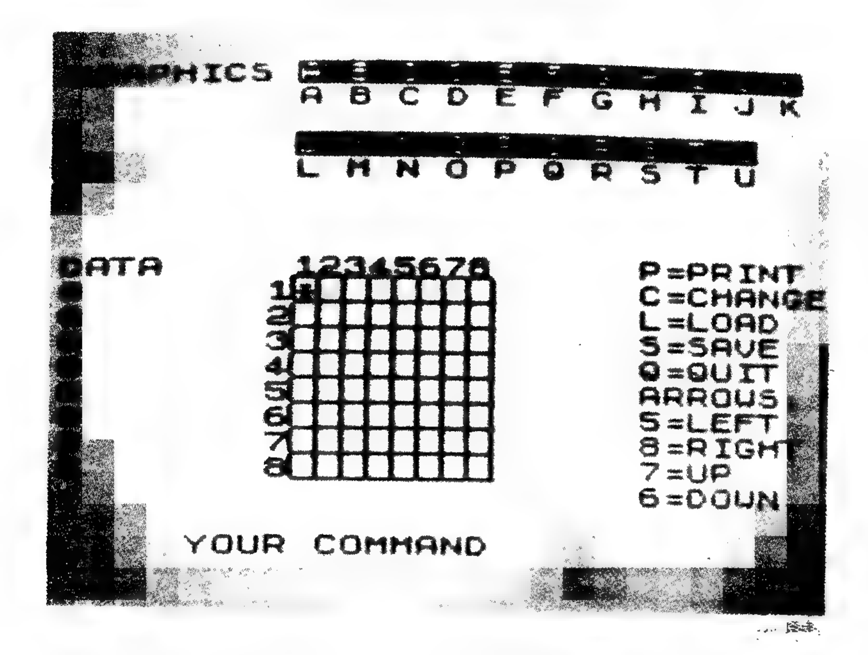
Recuperarlo es igual de fácil. Simplemente escriba:

LOAD "nombre" CODE USR "a"

o para el juego entero:

LOAD "caracteres" CODE USR "a"

El número de bytes después de la dirección en LOAD no es necesario porque en la cinta hay un número fijo de bytes. Es aconsejable que el nombre del programa sea indicativo del contenido de los caracteres.




```
  2 REM GENERADOR DE CARACTERES
 40 BORDER 1: PAPER 5: INK 1: CLS
 50 POKE 23609,100
 60 PRINT AT 9,23;"P=IMPRIME";AT 10,23;
"C=CAMBIA";AT 11,23;"L=CARGA";AT 12,23;"
S=GUARDA"
 70 PRINT AT 13,23;"Q=FUERA";AT 14,23;"
FLECHAS";AT 15,23;"5=IZQ.";AT 16,23;"8=D
ER.";AT 17,23;"7=ARRIBA";AT 18,23;"6=ABA
JO"
 80 DIM G$(8,8)
 90 PRINT AT 1,0;"GRAFICOS"; INVERSE 1;
AT 1,9;"A B C D E F G H I J K"
```

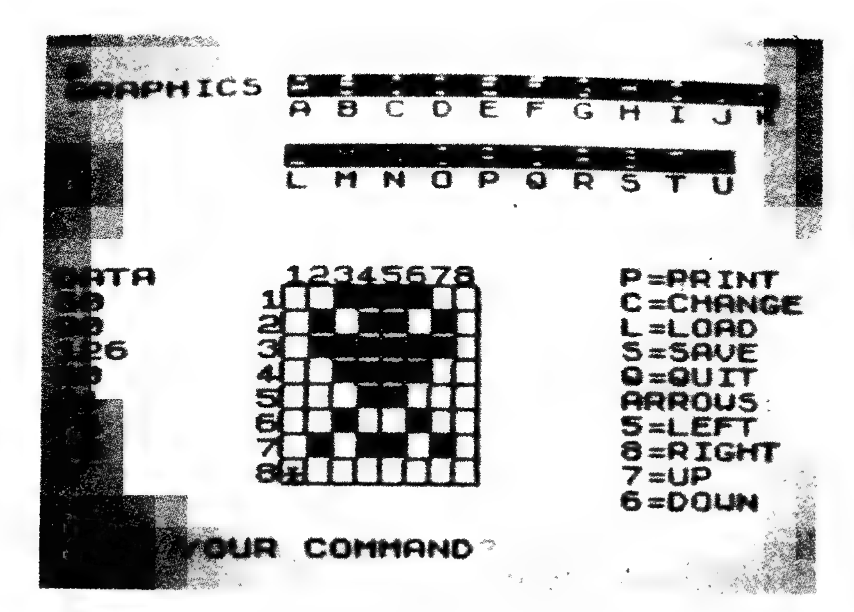
```
 95 REM LA LINEA 100 ES EN MODO
 GRAFICOS
 100 PRINT AT 2,9;"a b c d e f g h i j k
 "
 110 PRINT  INVERSE 1;AT 4,9;"L M N O P
Q R S T U"
 115 REM LA LINEA 120 ES EN MODO
GRAFICOS
 120 PRINT AT 5,9;"l m n o p q r s t U "
 130 GO SUB 540: REM PARRILLA DE CHR$
 140 FOR A=10 TO 17
 150 FOR B=9 TO 17
 160 PRINT  OVER 1;AT B,A-1;"_";AT A,B;"
u"
 170 LET S=A-9: PRINT  OVER 1;AT 9,A-1;S
;AT A,8;S
 180 NEXT B: NEXT A: OVER 0
 190 LET X=10: LET Y=9
 200 PRINT  OVER 1;AT X,Y;"*"
 210 PAUSE 1
 220 PRINT AT 9,0;"DATA"
 230 FOR A=10 TO 17: PRINT AT A,0;VAL ("
BIN "+G$(A-9));"    ": NEXT A
 240 PRINT AT 20,7;"SU COMANDO"
 250 PRINT  OVER 1;AT 20,17;"?": OVER 0
 260 IF INKEY$="6" THEN  GO TO 360
 270 IF INKEY$="7" THEN  GO TO 400
 280 IF INKEY$="c" THEN  GO TO 620
 290 IF INKEY$="5" THEN  GO TO 440
 300 IF INKEY$="s" THEN  GO TO 680
 310 IF INKEY$="p" THEN  GO TO 990
 320 IF INKEY$="8" THEN  GO TO 480
 330 IF INKEY$="1" THEN  GO TO 760
 340 IF INKEY$="q" THEN  STOP
 350 GO TO 210
 360 PRINT  OVER 1;AT X,Y;"*": LET X=X+1
: IF X=18 THEN  LET X=17
 370 PAUSE 10
 380 PRINT  OVER 1;AT X,Y;"*"
 390 GO TO 210
 400 PRINT  OVER 1;AT X,Y;"*": LET X=X-1
: IF X=9 THEN  LET X=10
 410 PAUSE 10
 420 PRINT  OVER 1;AT X,Y;"*"
```

```
 430 GO TO 210
 440 PRINT  OVER 1;AT X,Y;"*": LET Y=Y-1
: IF Y=19 THEN  LET Y=9
 450 PAUSE 10
 460 PRINT  OVER 1;AT X,Y;"*"
 470 GO TO 210
 480 PRINT  OVER 1;AT X,Y;"*": LET Y=Y+1
: IF Y>16 THEN  LET Y=16
 490 PAUSE 10
 500 PRINT  OVER 1;AT X,Y;"*"
 510 GO TO 210
 520 STOP
 530 REM
 540 REM PARRILLA DE CARACTERES
 550 REM
 560 FOR A=0 TO 7: POKE USR "U"+A,128: N
EXT A
 570 FOR A=1 TO 8
 580 LET G$(A)="00000000"
 590 NEXT A
 600 RETURN
 610 REM
 620 REM CAMBIO DEL BIT CHR$
 630 PRINT AT 0,0;"p": REM
 640 LET G$(X-9,Y-8)=("1" AND G$(X-9,Y-8
)="0")+("0" AND G$(X-9,Y-8)="1")
 650 PRINT  INK 0; OVER 1;AT X,Y;" "
 660 GO TO 210
 670 REM
 680 REM GUARDA CHR$
 690 REM
 700 INPUT "QUE CHR$?  "; LINE C$
 710 FOR A=0 TO 7: POKE VAL ("USR "+"C$"
)+A,VAL ("BIN "+G$(A+1)): NEXT A
 715 REM LA LINEA 720 ES EN
 GRAFICOS
 720 PRINT AT 2,9;"a b c d e f g h i j k
"
 725 REM LA LINEA 730 ES EN
 GRAFICOS
 730 PRINT AT 5,9;"l m n o p q r s t u"
 740 GO TO 210
 750 REM
 760 REM CARGAR CHR$
```

```
 770 REM
 780 INPUT "QUE CRH$?  "; LINE C$
 790 FOR W=1 TO 8
 800 LET A=PEEK ((USR C$)+W-1)
 810 LET T=128
 820 FOR S=1 TO 8
 830 LET G$(W,S)="0"""
 840 IF A>=T THEN  GO SUB 880
 850 LET T=T/2
 860 NEXT S
 870 NEXT W: GO TO 910
 880 LET G$(W,S)="1"
 890 LET A=A-T
 900 RETURN
 910 FOR A=1 TO 8
 920 NEXT A
 930 FOR W=1 TO 8
 940 FOR S=1 TO 8
 950 IF ATTR (W+9,S+8)=40 THEN  PRINT  O
VER 1;AT W+9,S+8;" "
 960 IF G$(W,S)="1" THEN  PRINT  INK 0;
OVER 1;AT W+9,S+8;" "
 970 NEXT S: NEXT W: GO TO 210
 980 REM
 990 REM CHR$ PARA PINTAR
1000 REM
1010 LPRINT CHR$ (CODE C$+47)
1020 LPRINT "DATA:"
1030 FOR A=1 TO 8
1040 LPRINT G$(A),VAL ("BIN "+G$(A))
1050 NEXT A
1060 GO TO 210
```

IDEAS PARA MODIFICAR EL PROGRAMA

Hay cantidad de modificaciones que usted puede hacer para conseguir que este programa sea aún más versátil. Lo primero de todo es que contiene muy pocas rutinas que comprueben si las respuestas dadas por el usuario son correctas o no. Por ejemplo, puede añadir una rutina que compruebe que el carácter que responde a la pregunta "Qué carácter?" sea una letra entre la A y la U (simplemente ponga INPUT A$, IF A$ > "U" THEN PRINT A$; "no es un gráfico definible").



También puede construir rutinas que hagan girar el tablero, lo que le será muy útil para juegos de naves espaciales. Por último puede crear un formato de pantalla diferente, en donde haya cuatro tableros de 8 × 8 formando una grande de 16 × 16. Entonces puede tener la opción de crear un carácter en cualquiera de los tableros. De este modo puede fabricar caracteres más grandes y ver cómo quedan sin necesidad de correr el programa que los use.

OVER

El Spectrum dispone de un comando muy útil llamado **OVER,** que ejecuta lo que se llama un OR exclusivo en la impresión. Esto significa que cuando un carácter se imprime en la misma posición que otro, si OVER está puesto (OVER 1), entonces en vez de borrarse el primero, se produce la fusión de los dos. Para entender mejor cómo funciona OVER necesitamos volver de nuevo a la numeración binaria. Los dos bytes de cualquier posición de la pantalla, son comparados bit a bit, de manera que dos unos o dos ceros dan un cero, y si son distintos dan un uno.

		A Ø	A 1
B	Ø	Ø	1
B	1	1	Ø

Obviamente, OVER es otra de las ampliaciones que le podría hacer al programa de construcción de caracteres. Necesitaría añadir una subrutina que comparara dos caracteres bit a bit y pintara un cuadrado blanco si son iguales, y uno negro si son distintos.

El OR exclusivo de estos dos números binarios da:

```
10011010
01010001
--------
11001011
```

Como puede ver es posible crear un carácter, entonces crear otro, y obtener la simulación de OVER en un tablero mayor.

MÁS DE 21 GRÁFICOS DEFINIBLES

Hasta ahora, sólo me he referido a los 21 caracteres para los que el Spectrum tiene reservada un área de memoria que va desde la dirección USR "A" hasta USR "U" + 8; sin embargo, si usted lo desea puede redefinir la totalidad del juego de caracteres (con excepción de los gráficos que se encuentran en las teclas numéricas).

Para ver esto instantáneamente haga (pero cuidado: todos los caracteres se convertirán en un grupo aleatorio de puntos; asegúrese de que no tiene ningún programa vital en memoria):

POKE 23607,0 (ENTER)

Si tiene algún programa en memoria verá que el listado se ha convertido en un barullo de puntos. Para volver a la situación normal pulse ENTER (para asegurarse de que tiene el cursor K, aunque no pueda reconocerlo) y pulse la tecla 0 (POKE) cuidadosamente seguida de 23607,60. ¡Es mejor que no mire la pantalla mientras lo hace! Ahora pulse ENTER y la situación volverá a su cauce normal, si esto no ocurre, no desespere, simplemente desenchufe y vuelva a enchufar.

Hay una variable llamada CHARS, que se encuentra en

las posiciones de memoria 23606 y 23607. En estas posiciones, se encuentran los bytes que le dicen al Spectrum la dirección (menos 256) de la tabla de caracteres en la ROM. Puede ver cuál es este número escribiendo:

PRINT PEEK 23606 + 256* PEEK 23607

Obtendrá la respuesta 15360. Lo que le he hecho hacer hace un momento es cambiar el byte de la posición 23607 por un 0 y como el otro también es 0 (compruébelo con PEEK) su Spectrum ha creído que la tabla de caracteres empezaba en la posición 256. Habrá adivinado que cambiando, esta dirección en múltiplos de 8 se pueden obtener efectos sorprendentes. Escriba esto (teniendo el juego normal):

POKE 23606,8

Es interesante, ¿verdad?

Todo esto se puede usar para hacerle más difícil a la otra gente, el leer los listados de su programa. Sin embargo, su programa, también usará los caracteres, entonces tendrá que definir un nuevo juego, como máximo de 21 que no serán afectados al hacer POKEs en las variables del sistema.

Una aplicación más seria consiste en reemplazar el juego de caracteres del Spectrum, por el suyo propio. La manera más fácil de hacerlo es reservar un área al final de la memoria usando CLEAR. Como el mismo nombre sugiere (CLEAR en inglés significa "limpiar") este comando borra toda la memoria a partir de la dirección que se indica tras él. Así por ejemplo, CLEAR 30000 reserva espacio desde la posición 30000 en adelante. Dependiendo del tamaño de la memoria de su Spectrum (16K o 48K) usted necesitará hacer CLEAR seguido de 32767 — x o bien de 65535— x donde x es el número de caracteres que va a definir multiplicado por 8. Después de esto, necesita colocar los datos (bytes) de sus caracteres en esta área de memoria, usando POKE. Si usted usa un programa anterior (el constructor de caracteres), para obtener los bytes de los nuevos, simplemente requerirá una rutina como ésta para hacer todo el trabajo:

```
10 FOR a = 32000 TO 32000 + 8*95    (esto es para el
20 INPUT n                          juego entero y con
30 POKE a,n                         Spectrum de 16K)
40 NEXT a
```

Por supuesto que como alternativa para cometer menos errores, puede colocar todos los bytes (que aquí se cargan a la variable n) en una línea DATA y utilizar READ en la línea 20.

SONIDO EN SU SPECTRUM

Vimos en la primera parte que en el Spectrum, el sonido se produce con el comando BEEP seguido de dos números, el primero de los cuales indica la longitud y el segundo el tono. Por ejemplo:

```
BEEP 0.5, 10
```

Todo esto no parece muy excitante, sin embargo, es posible crear sonidos mucho más interesantes y útiles, con ayuda del BASIC, y si usted quiere intentar algo de código máquina (véase la última parte), podrá obtener sonidos asombrosos.

Con ayuda del BASIC vemos todas las notas que se pueden obtener:

```
1 FOR N = -60 TO 69
20 BEEP 0.2 , N
30 NEXT N
```

O de un modo más aleatorio:

```
10 REM MUSICA ALEATORIA
20 LET nota = (RND* 100) — 50
30 LET nota = RND
40 BEEP dur, nota
50 GOTO 20
```

Observará que el sonido del Spectrum es bastante bajo, pero usted puede amplificarlo usando el cassette o un amplificador. La señal sale tanto de MIC como de EAR, quizás un poco más fuerte por este último, pruebe usted

mismo cuál es la salida que le va mejor para su amplificador. Cuando el Spectrum no envía sonido al amplificador, éste emite un zumbido bastante molesto. El único modo de evitarlo es mediante filtros o bien encontrar un punto crítico de volumen en el que se oiga suficiente el sonido pero no el zumbido.

Otro problema al amplificar el sonido es que el "clic" del teclado también queda amplificado. Pero el volumen del "clic" se puede alterar gracias a otra variable del sistema. Escriba:

POKE 23609, 100

Esto producirá un sonido más alto. Se puede "pokear" cualquier número entre 0 y 255. Pero si este número es muy grande, el "clic" se enlentece mucho. Yo encuentro que los valores alrededor de 100 son los más ideales.

Obviamente, mientras los enfuches MIC y EAR están conectados al amplificador, no pueden usarse para LOAD y SAVE. Un interruptor-conmutador sería una buena solución, si no quiere estar enchufando y desenchufando.

Volvamos a la música. Detrás de esta numeración de notas, hay por supuesto una regla. Así la nota 0 corresponde al DO central, el 1 es el DO sostenido, el 4 el MI, etc. Si sabe algo de música verá que hay doce números en cada octava, por lo tanto el 12 es el DO una octava mayor que el central. Usando esto junto con lo que ya sabemos de manipulación de cadenas, podemos construir un programa que permita introducir las notas directamente en lugar de estar escribiendo largos comandos BEEP. Aquí está, se llama COMPOSER:

```
  10 REM COMPOSITOR
  20 DIM X(50): DIM Y(50)
  30 LET K=0: LET L=1
  40 BORDER 0: PAPER 6: INK 9: CLS
  50 PRINT AT 0,10; INVERSE 1;"EL COMPOS
ITOR"''
  60 PRINT "ESCRIBE LA MUSICA EN FORMA D
E LINEA CON NUMEROS Y LETRAS"
  70 PRINT '''PON LAS NOTAS EN LETRAS SE
GUIDAS POR LA DURACION EN TIEMPOS"
  80 PRINT '''TIENES DOS OCTAVAS, LA BAJ
```

```
A DESDE LA A A LA G Y LA ALTA DESDE LA a
 A LA g"
  90 INPUT N$
 100 FOR A=1 TO LEN N$ STEP 2
 110 IF CODE N$(A)<97 THEN  GO TO 280
 120 IF N$(A)="a" THEN  LET K=-0.5
 130 IF N$(A)="d" THEN  LET K=0.5
 140 IF N$(A)="e" THEN  LET K=1
 150 IF N$(A)="f" THEN  LET K=1
 160 IF N$(A)="g" THEN  LET K=1.5
 170 LET Y(L)=(CODE N$(A)-87)+(2*K)
 180 LET L=L+1
 190 LET K=0
 200 NEXT A
 210 LET L=1
 220 FOR T=2 TO LEN N$ STEP 2
 230 LET X(L)=VAL N$(T)/2
 240 LET L=L+1
 250 NEXT T
 260 GO TO 360
 270 STOP
 280 IF N$(A)="A" THEN  LET K=-0.5
 290 IF N$(A)="B" THEN  LET K=-0.5
 300 IF N$(A)="F" THEN  LET K=0.5
 310 IF N$(A)="G" THEN  LET K=0.5
 320 LET Y(L)=(CODE N$(A)-67-K)*2
 330 LET K=0
 340 LET L=L+1
 350 GO TO 200
 360 FOR Z=1 TO LEN N$/2
 370 BEEP X(Z)/2,Y(Z)
 380 NEXT Z
 390 STOP
```

Como podrá observar al ejecutar este programa, le permite escribir las letras de la A a la G para una octava baja en la que la C representa al Do central, las letras de la 'a' a la 'g' son una octava más alta. He escogido el número 1 para representar un tiempo, así una melodía podría ser "A1 B1 C2 A1 C2". Un tiempo tiene una duración de

N.T. Las notas se escriben en nomenclatura anglosajona, es decir son C, D, E, F, G, A, B en lugar de Do, Re, Mi, Fa, Sol, La, Si; lo que por otro lado facilita mucho la manipulación de las cadenas.

0.25; si usted quiere ir más rápido o más lento, entonces esta constante puede ser establecida por usted. Además, ¿por qué no añadir una línea que pregunte la velocidad de ejecución?

5 INPUT "Qué compás? 1 lento, 5 rápido"; d

Hacer música con este programa es mucho más fácil que usar los interminables BEEPs y también más fácil que escribir primero la partitura en un papel. Aquí, la melodía, se guarda en una cadena, así que si usted quiere guardar su melodía en el cassette puede grabar todo el programa entero, pero luego al hacer LOAD ejecútelo con GOTO 1 en lugar de RUN, pues este último comando borra las variables.

¿Por qué no convertir el Spectrum en un órgano? Este programa utiliza el comando INKEY$ para conseguirlo. INKEY$ da como resultado una cadena conteniendo la tecla que se está pulsando en aquel momento y según su resultado, se ejecuta una nota u otra. He puesto el Do central en la tecla T y con tres octavas a elegir. Las teclas numéricas son los sostenidos y bemoles, las teclas de la Q a la P corresponden a las teclas blancas del órgano. También puede accionar un vibrator.

```
   2 REM ****ORGANO****
   3 BORDER 0: PAPER 2: INK 7: CLS
   4 PRINT AT 8,0;"PULSE 'Z' PARA OCTAVA
 ALTA,           'X' PARA LA BAJA"
   5 PRINT AT 16,0; INVERSE 1;"PULSE 'V'
 PARA VIBRATOR,              'B' PARA SA
LIR DE VIBRATOR"
   6 PRINT AT 19,0; INVERSE 1;"PULSE 'C'
 PARA LA OCTAVA CENTRAL    LA TECLA 'T'
ES EL DO        "
   7 LET k=0: LET x=0.3
   8 PAUSE 500
   9 REM TECLADO VISUALIZADO
  10 CLS
  11 PRINT  INVERSE 1;AT 10,4;"Q";AT 10,
6;"W";AT 10,8;"E";AT 10,10;"R";AT 10,12;
"T";AT 10,14;"Y";AT 10,16;"U";AT 10,18;"
I";AT 10,20;"O";AT 10,22;"P"
```

```
  15 PRINT  PAPER 0;AT 8,5;"2";AT 8,7;"3
";AT 8,9;"4";AT 8,13;"6";AT 8,15;"7";AT
8,19;"9";AT 8,21;"0"
  18 REM debe tocarse con
minusculas
  20 IF INKEY$="z" THEN  LET K=12
  21 IF INKEY$="t" THEN  BEEP x,0+k
  22 IF INKEY$="6" THEN  BEEP x,1+k
  23 IF INKEY$="y" THEN  BEEP x,2+k
  24 IF INKEY$="7" THEN  BEEP x,3+k
  25 IF INKEY$="u" THEN  BEEP x,4+k
  26 IF INKEY$="i" THEN  BEEP x,5+k
  27 IF INKEY$="9" THEN  BEEP x,6+k
  28 IF INKEY$="o" THEN  BEEP x,7+k
  29 IF INKEY$="0" THEN  BEEP x,8+k
  30 IF INKEY$="p" THEN  BEEP x,9+k
  31 IF INKEY$="r" THEN  BEEP x,-1+k
  32 IF INKEY$="4" THEN  BEEP x,-2+k
  33 IF INKEY$="e" THEN  BEEP x,-3+k
  34 IF INKEY$="3" THEN  BEEP x,-4+k
  35 IF INKEY$="w" THEN  BEEP x,-5+k
  36 IF INKEY$="2" THEN  BEEP x,-6+k
  37 IF INKEY$="q" THEN  BEEP x,-7+k
  38 IF INKEY$="x" THEN  LET k=-12
  39 IF INKEY$="c" THEN  LET k=0
  40 IF INKEY$="v" THEN  LET x=0.03
  41 IF INKEY$="b" THEN  LET x=0.3
  42 GO TO 20
```

Usted probablemente pensará en muchos refinamientos para este programa. ¿Por qué no tener al menos dos teclados, como los órganos grandes? Puede tener las hileras de arriba para las octavas altas y las de abajo para las bajas. ¿Por qué no añadir también un tempo? Puede añadir una rutina que con sólo pulsar una tecla produzca un "BUM CHA CHA" en lugar de un solo sonido. El Spectrum también permite variaciones fraccionarias de tono. Este hecho puede ser usado para afinar su órgano con otro instrumento. Quizá pueda añadir una rutina de afinación como ésta:

```
9990 IF INKEY$ = "N" THEN LET d = d + 0.05
9991 IF INKEY$ = "M" THEN LET d = d - 0.05
```

(Donde d representa el tono de las notas.)

Por supuesto que también puede apartarse de la práctica usual y crear un instrumento con octavas de 16 notas o incluso más, tal como ocurre en la música oriental.

Por último, le sugiero que para tocar este órgano, conecte su Spectrum a un amplificador. El sonido es mucho mejor cuando se amplifica. Por supuesto que con un poco de imaginación puede convertir este programa para que él mismo componga sus melodías. También puede usar los caracteres definibles por el usuario para crear las partituras de las notas y conseguir que la nota correcta, con la longitud correcta, aparezca en la pantalla sobre un pentagrama mientras usted toca. Quizás es un proyecto muy ambicioso, pero aquí tiene a PARTITURA un programa que es efectivamente una extensión del anterior:

```
  10 REM PARTITURA
  20 PAPER 5: INK 0: BORDER 1: CLS
  30 LET A=151
  40 REM DIBUJA PENTAGRAMA
  50 FOR B=1 TO 5
  60 PLOT 0,A: DRAW 255,0
  70 LET A=A-8
  80 NEXT B
  90 FOR A=0 TO 7
 100 READ T
 105 REM LO DE ENTRE COMILLAS ES
 'A' EN GRAFICOS
 110 POKE USR "a"+A,T
 120 NEXT A
 130 FOR B=0 TO 7
 140 READ T
 145 REM LO DE ENTRE COMILLAS ES
 'B' EN GRAFICOS
 150 POKE USR "b"+B,T
 160 NEXT B
 170 PRINT  OVER 1;AT 3,3;"a";AT 4,3;"b"
:
 175 REM A I B SON GRAFICOS
 180 BEEP 0.25,0
 190 REM PARTE SUPERIOR DE LA
NOTA
 200 DATA 0,4,7,5,4,4,4,4
```

```
210 REM PARTE INFERIOR DE LA
NOTA
220 DATA 0,60,124,124,124,56,0,0
230 STOP
```

Usando el comando BEEP es difícil crear en BASIC sonidos raros para juegos, dado que sólo se permite variar la duración y el tono. Sin embargo, algunas rutinas muy útiles permiten crear sonidos sin estas limitaciones. Usted puede crear sonidos del tipo de un paso. Por ejemplo, aquí tenemos a un hombre subiendo una escalera:

```
  2 REM SUBIENDO UNA ESCALERA
  3 REM creando un hombre CHR$
  4 BORDER 4: PAPER 6: CLS
 10 POKE USR "p",  BIN 00011000
 20 POKE USR "p"+1,BIN 00100100
 30 POKE USR "p"+2,BIN 10011001
 40 POKE USR "p"+3,BIN 01111110
 50 POKE USR "p"+4,BIN 00011000
 60 POKE USR "p"+5,BIN 01100100
 70 POKE USR "p"+6,BIN 10000100
 80 POKE USR "p"+7,BIN 00000100
 90 REM Segundo hombre para movimiento
100 POKE USR "u",  BIN 00011000
110 POKE USR "u"+1,BIN 00100100
120 POKE USR "u"+2,BIN 10011001
130 POKE USR "u"+3,BIN 01111110
140 POKE USR "u"+4,BIN 00011000
150 POKE USR "u"+5,BIN 00100110
160 POKE USR "u"+6,BIN 00100001
170 POKE USR "u"+7,BIN 00100000
200 REM Dibuja la escalera
210 LET x=BIN 01000010
220 POKE USR "a",x
230 POKE USR "a"+1,x
240 POKE USR "a"+2,BIN 01111110
250 POKE USR "a"+3,x
260 POKE USR "a"+4,x
270 POKE USR "a"+5,x
280 POKE USR "a"+6,BIN 01111110
290 POKE USR "a"+7,x
300 FOR t=0 TO 21
```

```
 310 PRINT AT t,10;"a": REM Este es el g
rafico CHR$ en la tecla 'A'
 320 NEXT t
 330 LET x=0: LET y=21: LET z=-1
 340 FOR n=y TO x STEP z
 350 PRINT AT n,10;"p": PAUSE 2-z: PRINT
 AT n,10;"u"
 355 PAUSE 2-z
 360 REM Recuerda que son graficos CHR$!
 370 PRINT AT n,10;"a"
 380 IF ABS z=z THEN  GO TO 500
 390 BEEP 0.02,30: BEEP 0.02,40
 400 NEXT n
 410 LET k=x: LET x=y: LET y=k: LET z=-z
 420 GO TO 340
 500 BEEP 0.01,(40-(n/2))
 510 NEXT n: GO TO 330
```

También hay sonidos muy variados para programas de invasores:

```
10 REM sonidos especiales
20 FOR x=-10 TO 0
30 BEEP 0.0125,x
40 NEXT x
50 FOR y=0 TO -5 STEP -1
60 BEEP 0.0125,y
70 NEXT y
```

```
10 REM sonidos especiales
20 LET d=0.0125
30 FOR x=1 TO 2
40 FOR y=4 TO 16 STEP 2
50 BEEP d,y
60 NEXT y
70 NEXT x
```

También es posible tener sonidos de nave espacial:

```
 2 REM  nave espacial
 5 LET n=0
10 FOR a=0 TO 3
20 BEEP .01,n
```

```
30 LET n=n+3
40 IF n>60 THEN  GO TO 5
50 GO TO 10
```

Combinar gráficos y sonido, puede ser divertido:

```
 10 REM  collage musical
 20 INK 7: BORDER 1: PAPER 1: CLS
 30 FOR x=0 TO 31
 40 LET n=RND*20
 50 LET d=RND*0.4
 60 LET i=(RND*6)+1
 70 PRINT  INK i;AT 20-n,x;"■"
 80 BEEP d,n
 90 NEXT x
100 CLS : GO TO 30
```

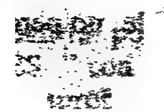
En ordenadores más sofisticados, usted puede tener otros tipos de control, además de estos, sobre el sonido producido. No sólo puede definir la duración y el tono, sino también cómo varían estos factores en el tiempo. A esta variación se le llama envolvente. Podemos simular algo de esto con el Spectrum, teniendo en cuenta que tanto la duración como el tono pueden tomar valores fraccionarios:

```
 10 REM frecuencia/duracion
 20 INK 7: BORDER 1: PAPER 1: CLS
 30 LET d=0.05
 40 FOR f=0 TO 2 STEP 0.5
 50 BEEP d,f
 60 DRAW 300*d,20*f
 70 LET d=d-0.005
 80 NEXT f
 90 DRAW 50,0: BEEP 0.5,2
100 GO TO 10
```

Para añadir aún algo más a estas rutinas, puede hacer que los cambios que se hacen en la duración y el tono varíen asimismo según el tiempo. Así al principio, la frecuencia empieza a crecer según una razón dada, y enseguida crece el tono a una razón más rápida. De todas maneras, las restricciones del comando BEEP no permiten crear sonidos

aceptables en este sentido. También es posible crear zumbidos de "clics" pero que deben de tener muy poco uso.

Ahora bien, si usted va a intentar hacer algo de código máquina podrá crear sonidos que impresionan mucho más, pero de esto ya hablaremos más tarde.



Tercera Parte

Juegos



CAPÍTULO SÉPTIMO

Los juegos que juega la gente

Escribir buenos juegos en un ordenador, es indiscutiblemente un arte. Por eso es bastante difícil conseguirlo, leyendo simplemente unas pocas páginas de un libro. Sin embargo, espero dar aquí los consejos necesarios para colocarle en el buen camino.

En general se pueden programar dos tipos de juegos. Los primeros se basan en el movimiento de gráficos, y en ellos son muy importantes las rutinas para mover caracteres en la pantalla, dibujos bien hechos, etc. El segundo tipo es el juego de aventuras, donde las preguntas y el análisis de las respuestas están al orden del día. En el primer tipo también tienen mucha importancia los trucos para aumentar la velocidad del juego (especialmente si se escribe en BASIC); mientras que el segundo tipo, necesita de una planificación lógica, profunda, para que no se convierta en un juego tonto y simple. A estos dos tipos de juegos, los llamaremos respectivamente, juegos gráficos y juegos verbales.

JUEGOS VERBALES

Hay que tener en cuenta dos aspectos importantes a la hora de escribir juegos verbales: por un lado está todo el proceso lógico que debe determinar en qué lugar se encuentra el jugador (por ejemplo, en qué parte del laberinto), y por otro, está la programación del orden que deben seguir los problemas o tesoros con los que se encuentra a lo largo del juego. También son de vital importancia aspectos como: los criterios para aceptar o deshechar una res-

puesta, hacer las preguntas lo más claras posible, no perder de vista las "posesiones", la puntuación, etc.

LAS RESPUESTAS

En la primera parte del libro ya vimos cómo se puede hacer para detectar si un jugador había respondido "YES" a una pregunta. Uno de los métodos, simplemente consistía en dimensionar un conjunto con un solo elemento [por ejemplo, R$ (1)] y luego mirar si la variable era igual a "Y". Este método no era muy bueno puesto que se dejaba algunas posibilidades, como por ejemplo, que el jugador incluya un espacio (por error) al responder, o que su respuesta empiece por "y" pero que no signifique "YES".
El primer problema se solucionó con una rutina que buscaba las letras "YES" en cualquier lugar de la cadena que se daba como respuesta. Lo que no se incluyó para no complicar el programa es el hecho de que las letras Y E S pueden formar parte de una palabra más larga que no significa YES. Esto se puede solucionar comprobando que las letras YES se encuentran rodeadas de espacios. Aquí hay una manera de hacerlo aunque posiblemente no es la más elegante. Nota: asegúrese de que tiene puestas las mayúsculas (CAPS LOCK):

```
10 INPUT "OTRA VEZ?"; R$
20 IF R$ = "YES" THEN GOTO 100
30 IF LEN R$<3 THEN GOTO 80
40 IF LEN R$ = 4 THEN GOTO 200
50 FOR X = 2 TO LEN R$ - 1
60 IF R$(X TO X + 4) = " YES " THEN GOTO 100
70 NEXT X
80 PRINT "ADIOS": STOP
90 STOP
100 PRINT "DE ACUERDO, CONTINUO"
110 GOTO . . . (programa)
200 IF R$ ( TO 3) = "YES" THEN GOTO 100
210 IF R$ (2 TO) = "YES" THEN GOTO 100
220 GOTO 80
```

Esto se podría mejorar añadiendo un espacio al principio y al final de la respuesta, porque si la palabra YES se

encuentra por ejemplo al principio, la comparación de "YES " con " YES " daría negativa.

Comprobar si el jugador ha respondido sólo con letras es muy sencillo, usando el hecho de que las cadenas se pueden comparar así:

```
10 INPUT "Por dónde vas?", D$
20 FOR N = 1 TO LEN D$
30 IF D$(N)>"Z" OR D$(N)<"A" THEN GOTO 10
40 NEXT N
```

Pero hay que tener cuidado, ya que el Spectrum diferencia entre mayúsculas y minúsculas, y para tener esto en cuenta habría que ampliar así la línea 30:

```
30 IF CODE D$(N)>122 OR CODE D$(N)<65
   THEN GOTO 10
```

Esto comprueba todos los caracteres exceptuando los gráficos cuyos códigos van del 91 al 95 (corchetes, flecha, etc.). Pero se puede considerar esto suficientemente seguro, aunque si quiere puede añadir lo necesario para evitar que se introduzcan estos caracteres. Las entradas numéricas también se pueden comprobar usando CODE:

```
10 INPUT N$
20 FOR X = 1 TO LEN N$
30 IF CODE N$(X)>57 OR CODE N$(X)<48 THEN
   GOTO 10
40 NEXT X
50 LET N = VAL N$
```

Observe cómo uso INPUT para entrar los números como cadenas, y cuando han superado la comprobación se convierte la cadena en un número, usando VAL. Sin embargo, hay un problema al hacer INPUT con cadenas, y es que las comillas se imprimen después de INPUT y pueden ser borradas por accidente. Además, cualquier persona podría ver el listado de su programa, fácil y simplemente, borrando la comilla de la izquierda y pulsando STOP. Felizmente, el Spectrum tiene una manera de solucionar esto, usando LINE:

```
10 INPUT "¿DE QUÉ MODO", LINE A$
```

Esto coloca el mensaje con el interrogante en la parte baja de la pantalla, pero no las comillas. Si usted ha hecho un buen trabajo filtrando las respuestas que pueden causar la detención del programa por error, entonces será muy difícil para el programa, y casi imposible que esto suceda por accidente.

Otra cosa que puede pasar es que alguien no conteste absolutamente nada, sino que como toda respuesta pulse la tecla ENTER. Para evitar que se tome como respuesta, añada la línea:

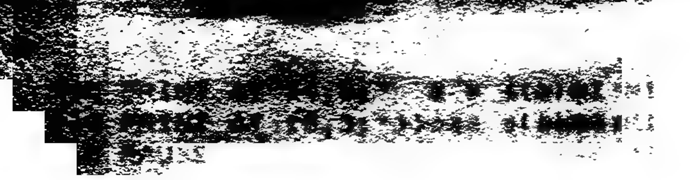
```
15 IF A$ = ""THEN GOTO 10
```

Observe que no hay espacios entre las dos comillas. Esto detecta cuando la respuesta es nula, y vuelve atrás para pedirla de nuevo.

También se puede detectar cuando alguien escribe 2*3 en lugar de 6. Esto se consigue utilizando este tipo de sentencias:

```
10 INPUT C$
20 IF C$ < > STR$ VAL C$ THEN PRINT
"Escríbalo de nuevo"
```

Esto funciona mientras que la persona que use el programa introduzca sólo combinaciones de números y símbolos, pero el programa se para con un mensaje de error en el momento de evaluar la función VAL si la cadena C$ contiene letras o símbolos no matemáticos.



CAPÍTULO OCTAVO
Programación de juegos

UN JUEGO DE LABERINTOS Y AVENTURAS

El juego que veremos ahora se llama LABERINTO y es en parte un juego de laberinto, y en parte un juego de aventuras en miniatura. La idea es empezar en la esquina inferior derecha del laberinto y trazarse un camino para llegar a la salida. Sin embargo, el mapa del laberinto no es siempre visible, y sólo puede verlo 5 veces en cada partida. A lo largo del camino se encontrará con tesoros y sufrirá peligros. Si usted no consigue el verdadero tesoro, o no sabe lo que hacer con él en el tiempo correcto, entonces pierde. Le sugiero que cuando escriba el programa repase varias veces el listado para corregir los posibles errores de transcripción. Ejecute el programa, y si éste detiene con algún error, mire la línea en que se ha producido y compárela con la que aquí tiene:

```
  2 REM *********************
  3 REM      EN EL LABERINTO
  4 REM *********************
 10 REM para ver el mapa entre
'MAP'
 25  LET F=0
 30 LET T=0
 35 GO SUB 1000
 40 LET A=0: RESTORE 110
 45 LET TRES=0
 50 LET X=31
 55 PAPER 2: INK 6: BORDER 6: CLS
 60 DIM Z$(36,6)
```

```
  65 DIM I$(6)
  70 DIM D$(1)
  75 DIM S$(1)
  80 FOR C=1 TO 36
  90 READ Z$(C)
 100 NEXT C
 110 DATA "000110","001010","000100","10
1010","000110","001010"
 120 DATA "100011","000111","001011","00
0111","001011","000001"
 130 DATA "000101","001011","000011","00
0011","010111","001000"
 140 DATA "000110","001101","011101","00
1011","000101","001010"
 150 DATA "000101","001010","000110","01
1111","001010","000011"
 160 DATA "000100","001001","000001","00
0101","001101","101001"
 170 INPUT "QUE CAMINO?(N,S,E,O)"; LINE
D$
 180 PRINT AT 0,0;"
          "
 195 IF TRES=1 AND D$="p" THEN  LET I$(1
+INT (X*6))="1"
 200 IF D$="m" THEN  GO TO 300
 210 IF d$="n" THEN  LET A=6
 220 IF D$="s" THEN  LET A=5
 230 IF D$="e" THEN  LET A=4
 240 IF D$="o" THEN  LET A=3
 250 IF X=18 AND A=4 THEN  GO TO 290
 260 GO SUB 500
 280 GO TO 170
 290 CLS : PRINT  INK 2; PAPER 6; FLASH
1;AT 10,10;"ESTAS FUERA": STOP
 300 REM MAPA: CLS
 310 PRINT AT 5,5;"*** *** ***"
 320 PRINT AT 6,5;"* * * * * *"
 330 PRINT AT 7,5;"* *** *** *"
 340 PRINT AT 8,5;"* * * * *"
 350 PRINT AT 9,5;"*** * * ***>>"
 360 PRINT AT 10,5;"  * * * *"
 370 PRINT AT 11,5;"******* ***"
 380 PRINT AT 12,5;"*     *   *"
 385 PRINT AT 13,5;"*** ***** *"
```

```
 390 PRINT AT 14,5;"  * * * * * "
 410 PRINT AT 15,3;">>*** * *****"
 415 PAUSE 0
 420 LET G=INT (X/6): LET H=X-1-(INT (X/
6))*6
 430 IF G<>(X/6) THEN  GO TO 450
 440 LET G=G-1: LET H=5
 445 PRINT AT 1,0;"X=";X;",";"G=";G;",";
"H=";H
 450 PRINT AT 5+2*G,5+2*H; OVER 1; FLASH
 1;"  "
 460 PAUSE 100: LET T=T+1
 470 IF T=5 THEN  GO TO 490
 480 CLS : GO TO 170
 490 CLS : PRINT AT 10,10; FLASH 1; INK
2; PAPER 6;"PERDISTE": STOP
 500 REM ESTE CAMINO?
 510 IF Z$(X,A)="0" THEN  GO TO 540
 520 IF Z$(X,A)="1" THEN  LET X=X+(1 AND
 A=4)-(1 AND A=3)+(6 AND A=5)-(6 AND A=6
)
 525 IF Z$(X,1)="1" THEN  GO TO 600
 526 IF Z$(X,2)="1" THEN  GO TO 700
 530 RETURN
 540 PRINT AT 0,0;"NO PUEDES IR POR AQUI
!": RETURN
 550 STOP
 600 REM TESORO
 610 PRINT AT 0,0;"PUEDES VER UN"+(V$ AN
D (INT (X/6))=1)+(L$ AND (INT (X/6))=0)+
(E$ AND (INT (X/6))=6)
 620 INPUT "QUE QUIERES HACER? "; LINE S
$
 630 IF S$="p" THEN  GO TO 640
 636 PRINT AT 0,0;"NO LO TIENES
           ": RETURN
 640 PRINT AT 0,0;"AHORA SI LO TIENES
           "
 645 LET TRES=1
 650 RETURN
 699 STOP
 700 REM PROBLEMA
 710 PRINT AT 0,0;"HAY UN"+(Q$ AND (INT
(X/6))=2)+(U$ AND (INT (X/6))=3)+(M$ AND
```

```
(INT (X/6))=4)
 720 INPUT "QUE HACES? "; LINE S$
 730 IF S$=(Y$(1) AND INT (X/6)=4)+(Y$(2
) AND INT (X/6)=3)+(Y$(3) AND INT (X/6)=
2) THEN GO TO 800
 740 PRINT AT 0,0;"LO SIENTO....
         ": GO TO 900+(RND*3)+1
 800 IF TRES>0 THEN GO TO 820
 810 PRINT "LO SIENTO, NO QUISO HACERLO.
...": STOP
 820 PRINT AT 0,0;"LE GUSTARIA QUE CONTI
NUARAS    ";AT 1,0;"
          "
 830 GO TO 170
 901 PRINT "TE HA GOLPEADO
   ": STOP
 902 PRINT "TE LO HA HECHO PAGAR MUY CAR
O  ": STOP
 903 PRINT "HAS CAIDO EN SU TRAMPA - ADI
OS ": STOP
1000 LET V$="DIAMANTE GIGANTE"
1010 LET L$="ELIXIR DE LA VIDA"
1020 LET E$="FRUTO DEL AMOR"
1030 LET Q$="DRAGON"
1040 LET M$="NIBELUNGO"
1050 LET U$="UNICORNIO"
1060 DIM Y$(3)
1070 LET Y$(1)="r": LET y$(2)="o": LET y
$(3)="t"
1090 RETURN
```

Aquí tiene un esquema que le indica lo que hace cada una de las partes de este programa:

LÍNEAS	COMENTARIO
25- 75	Definen las variables y dimensionan los conjuntos.
80- 160	Los datos para el laberinto se asignan a la cadena Z$.
170- 280	Se detecta la tecla pulsada.

300- 480	Dibuja el mapa en pantalla durante unos momentos.
500- 540	Detecta si la dirección respondida es posible.
600- 650	¿Qué tesoro ha encontrado?
700- 830	Un problema y su respuesta.
901- 903	Distintos finales.
1000-1090	Asignación de variables.

La clave para entender cómo funciona este programa la tenemos en las líneas de DATA de la 110 a la 160. Aquí está registrado el funcionamiento lógico del laberinto en forma de números binarios de seis bits, que le permiten saber al programa qué direcciones se pueden escoger en un punto determinado y también si hay un problema o un tesoro en esa posición. Esto se consigue dando un significado a cada uno de los seis bits:

bit 5	bit 4	biy 3	bit 2	bit 1	bit 0
tesoro	problema	oeste	este	sur	norte

Esto puede ser un poco complicado de entender a menos que se imagine que un 1 en una columna significa "sí"; así, por ejemplo, un 1 en el bit 5 significa que en esta posición hay un tesoro, un 1 en el bit 0 significa que usted puede ir en dirección norte desde aquí, etc. Por ejemplo:

> 010101 significa que se encuentra en una posición del laberinto en la que hay un problema y que desde aquí sólo puede ir al este o al norte.

El trabajo del programa consiste en ver si el bit 5 o el 4 contienen un 1, y si esto sucede ir a las rutinas de tesoro o problema respectivamente (vea las líneas 525 y 526 del programa). La otra tarea importante del programa consiste en interpretar las direcciones tecleadas en términos de los bits del 0 al 3. La manera más fácil de hacer esto es guardar los bits en una cadena y utilizar las grandes facilidades que ofrece el Spectrum para su tratamiento. Así, si

la dirección obtenida es "n" (norte), el programa lo único que hace es cargar en una variable (A) el valor 6 (ya que el bit 0, el que indica si se puede ir al norte, está en la sexta posición de la cadena) y comprobar si Z$ (X, A) es 1. Esto ocurre en las líneas 210-240 y 510-520.

Obviamente, si Z$ (X, A) es 0 entonces no puede ir hacia esa dirección, y la línea 540 se lo indica. La línea 520 es un poco más compleja ya que utiliza **AND.** Poniendo AND después de un campo y luego una condición, se provoca que dicho campo, sólo sea utilizado en el caso de que la condición sea cierta, y no se usa si la condición es falsa. Así:

PRINT ("Hola" AND 2 > 3)

Esto no imprimirá nunca "Hola" ya que la condición es falsa. Esto es lo que se usa en la línea 520 para hacer que su posición en el laberinto sea una más o una menos de la que era (es decir, ir al este o al oeste), o bien seis más o seis menos de lo que era (lo mismo que arriba o abajo). Un procedimiento similar se utiliza en la línea 610 para decidir con qué tesoro se ha encontrado, y otra vez en la 710 para decidir con qué problema se tendrá que enfrentar.

Cuando se encuentra con un problema o un tesoro, entonces el programa le pregunta qué desea hacer. Y sólo mira a la primera letra de la respuesta. Recordará que esto se puede hacer fácilmente dimensionando una cadena de un solo carácter de longitud. La ventaja de esto es que se evita el tener que poner tantas rutinas de comprobación de respuestas, para ver si la frase o la palabra introducida era la requerida o no. Por supuesto que cualquier respuesta con la primera letra correcta será tomada como tal (esto no debería decírselo a los jugadores).

Como puede ver, los tesoros y problemas son asignados a variables, al principio del juego, para que estén disponibles cuando se necesiten. En este caso he preferido dibujar el mapa directamente con PRINT AT (líneas 310-410) donde las posibles posiciones del aventurero son 36. La posición actual en el laberinto puede ser calculada a partir de X. Como puede ver en la línea 450 simplemente hay que añadir 5 al doble del valor de X para obtener la posición y hacer un FLASH en dicho punto. La línea 60 hace que el mapa sólo sea visible durante unos 2 segundos (PAUSE

100), y también almacena en la variable T el número de veces que se ha mirado el mapa, que si sobrepasan las cinco producen la pérdida de la partida. Con toda esta información usted está capacitado para averiguar cómo funciona el programa, especialmente si lo juega unas pocas veces.

ALGUNAS OBSERVACIONES

Hubiera sido muy fácil hacer un laberinto tridimensional, simplemente añadiendo dos bits más a las cadenas de DATA para indicar arriba o abajo. Las cadenas entonces tendrían una longitud de 8. El mapa dibujado, dependería de la planta en que estuviera.

El mapa se ha dibujado con PRINT AT por dos razones, una porque es más fácil, y otra porque así el mapa aparece en el listado. Sin embargo, al componer un juego como éste, en que la naturaleza y la forma del laberinto no deben ser vistas fácilmente, es mejor disfrazar un poco toda esta información. Los asteriscos que componen el mapa pueden ser registrados en líneas DATA como la posición en que se deben imprimir. Esto hará que sea un poco más difícil de descifrar. También es posible disfrazar los distintos tesoros y problemas almacenándolos en una variable como un conjunto de códigos y para leerlos, se utilizará el comando CHR$. Métodos similares se pueden usar para disimular el lugar en que se encuentran éstos.

Obviamente en este juego el laberinto es el mismo cada vez. Puede cambiarlo dibujándolo en un papel y entrando luego los correspondientes cambios tanto en las líneas DATA como en la rutina que lo dibuja. Sin embargo, pensando un poquito, sería posible crear un laberinto aleatorio, y de modo que el programa también decida en qué lugar se encontrarán los tesoros y los problemas. Para crear un laberinto lo que tiene que hacer esencialmente es crear 36 (para uno de 6 × 6 como éste) cadenas cada una compuesta por un número binario de 6 cifras. Esto se puede hacer asignando aleatoriamente un 1 o un cero a cada bit o bien crear también aleatoriamente números entre 1 y 63 y pasarlos luego a forma binaria, según una rutina que sugerí en la primera parte del libro. El mayor problema consiste en asegurarse que cada intersección conecta al menos con

otra, y también asegurar que es posible ir desde la entrada a la salida. Si se supera este obstáculo, se puede crear un programa muy interesante.

Por último, observará que cuando se pierde por no haber tratado correctamente un problema el programa termina con una frase aleatoria de las líneas 901-903. Esto se consigue en la línea 740 donde se toma la decisión de ir a un número de línea entre 901 y 903:

```
740 PRINT AT 0,0; "Lo siento...": GOTO
900 + (RND*3) + 1
```

Este método de escoger al azar la frase que se imprimirá causa un gran efecto en juegos en los que no se puede predecir la respuesta, con lo que se hacen menos aburridos. El método se puede usar en otro tipo de aplicaciones como en el siguiente programa FRASES:

```
  2 REM    ***************
  3 REM          frases
  4 REM    ***************
  5 REM CHERI DAVIS-LANGDELL:
      1982
  6 PAPER 1: INK 5: BORDER 1: CLS
 10 REM
 20 REM
 30 REM
 40 DIM a$(5,35)
 50 DIM b$(5,35)
 60 DIM c$(5,35)
 70 DIM d$(5,35)
 80 GO SUB 1000
 90 FOR n=1 TO 5
100 PRINT AT RND*10,0;a$(n)''b$(n)''c$(
n)''d$(n)
105 PAUSE 400
106 CLS
107 BEEP .2,20
110 NEXT n
140 LET n1=1+RND*4: LET n2=1+RND*4: LET
n3=1+RND*4: LET n4=1+RND*4
150 PRINT AT RND*10,0;a$(n1)''b$(n2)''c
$(n3)''d$(n4)
```

```
 160 PAUSE 400
 170 CLS : BEEP .2,20: GO TO 140
 990 STOP
1000 REM asignando lineas
1005 RESTORE 1120
1010 FOR x=1 TO 5
1020 READ a$(x)
1030 NEXT x
1040 FOR x=1 TO 5
1050 READ b$(x)
1060 NEXT x
1070 FOR x=1 TO 5
1080 READ c$(x)
1085 NEXT x
1090 FOR x=1 TO 5
1100 READ d$(x)
1110 NEXT x
1120 DATA "Me gusta apostar","El ansioso
 vive para jugar","Quien mal anda mal ac
aba","Olvida los problemas","Ven y rie c
onmigo"
1130 DATA "La primavera trae alegria","C
on junio, el verano","La fruta esta ya m
adura","Levantate antes que el sol","Tod
avia no se ha puesto el sol"
1140 DATA "Golpe a golpe, paso a paso","
Magico es el mar","El muerto yace bajo l
a luna","Que empiece el baile!","Al pan,
 pan y al vino, vino"
1150 DATA "El vagabundo ya no llorara ma
s","Va a llover, dulce pajarito","Quien
se mueve en la noche?","Pia gaviota, pia
","Poderoso caballero"
1160 RETURN
```

Otra variación en este tema se usó en el juego del ahorcado en la primera parte. Allí utilicé la idea de almacenar las posibles palabras a adivinar en líneas DATA, cada palabra con su propio número de línea. Luego, usando el hecho de que RESTORE n coloca el puntero para READ en la línea DATA que tiene por número n, sólo tuve que escoger n aleatoria y convenientemente.

Para que el programa se vea más elaborado, queda muy

bien el poner los mensajes finales en letras de doble alto. Aquí hay un programa que lo hace:

```
    2 REM *********************
    3 REM ESCRIBE EN DOBLE ALTO
    4 REM *********************
    5 LET l=0: LET r=-1
    6 BORDER 6: PAPER 6: CLS : LET z=0
    7 PRINT "USA SHIFT '0' PARA BORRAR,
   PULSA ENTER PARA NUEVA LINEA"
    8 PAUSE 0: CLS
    9 PAUSE 0: LET r=r+1: IF r=31 THEN  G
 O TO 1000
   10 LET t$=INKEY$
   15 IF t$="0" THEN  GO TO 500
   16 IF CODE INKEY$=12 THEN  GO TO 500
   17 IF CODE INKEY$=13 THEN  GO TO 1000
   20 FOR a=1 TO LEN t$
   30 FOR x=0 TO 6 STEP 2
   40 POKE USR "a"+x,PEEK (15616+(8*(CODE
  t$(a)-32))+z)
   50 POKE USR "a"+x+1,PEEK (15616+(8*(CO
 DE t$(a)-32))+z)
   55 LET z=z+1
   60 NEXT x
   70 FOR x=1 TO 7 STEP 2
   80 POKE USR "b"+x-1,PEEK (15616+8*(COD
 E t$(a)-32)+z)
   90 POKE USR "b"+x,PEEK (15616+8*(CODE
 t$(a)-32)+z)
   95 LET z=z+1
  100 NEXT x
  101 LET z=0
  105 REM  'A' y 'B' son graficos en la s
 iguiente linea
  110 PRINT AT l,r;"a";AT l+1,r;"b"
  120 NEXT a
  130 GO TO 9
  500 REM  BORRAR
  510 LET r=r-2
  520 GO TO 9
 1000 REM siguiente linea
 1010 LET l=l+2: LET r=-1: GO TO 9
```

El otro tipo de juego es el juego gráfico con movimiento. Star Gobbler y Colony son dos ejemplos de esta clase de juegos, así como lo son todos los juegos de invasores, marcianitos, etc.

Frecuentemente, estos juegos, requieren que el jugador mueva alguna cosa por la pantalla. Normalmente, esto se hace usando las flechas que están en las teclas 5 a 8 porque así ya se indica la dirección del movimiento. Aquí tenemos la rutina base:

```
10 IF INKEY$ = "6" THEN LET X = X + 1
20 IF INKEY$ = "7" THEN LET X = X - 1
30 IF INKEY$ = "5" THEN LET Y = Y - 1
40 IF INKEY$ = "8" THEN LET Y = Y + 1
50 GO TO 10
```

Aquí X representa la coordenada horizontal del objeto que está en la pantalla e Y representa a la coordenada vertical. Por supuesto que luego hay que imprimir el objeto en la pantalla, ya sea con PRINT AT o con PLOT, ya que esta rutina no contiene la impresión del objeto. Para que la figura que se imprime parezca realmente que se mueva, necesita usar alternativamente PRINT y PRINT OVER (o bien PLOT y PLOT OVER). Aquí tiene un ejemplo:

```
10 LET X = 10: LET Y = 15
20 IF INKEY$ = "7" THEN LET X = X - 1
30 IF INKEY$ = "6" THEN LET X = X + 1
40 IF INKEY$ = "8" THEN LET Y = Y + 1
50 IF INKEY$ = "5" THEN LET Y = Y - 1
60 PRINT AT X,Y;"*":PRINT AT X,Y;OVER 1;"*"
70 GOTO 20
```

Si introduce este programa y lo ejecuta, rápidamente observará dos problemas; primero que el asterisco puede exceder fácilmente los límites de la pantalla produciéndose un error ("Out of screen..."), y segundo que el asterisco se mueve muy deprisa y parpadea mucho. Estos problemas

se solucionan detectando los valores máximos de X e Y y añadiendo PAUSE entre PRINT y PRINT OVER.

```
25 IF X<0 THEN LET X = 0
35 IF X>20 THEN LET X = 20
45 IF Y>31 THEN LET Y = 31
55 IF Y<0 THEN Y = 0
60 PAUSE 2: PRINT AT X,Y;"*":PAUSE 2: PRINT
OVER1; AT X,Y ;"*"
```

Añadiendo estas líneas, se encontrará con que puede mover el asterisco hacia cualquier parte de la pantalla.

Hacer que el carácter sea una nave espacial, aunque contenga más de un cuadrado, y moverlo por la pantalla es igual de fácil. Pero recuerde que si la nave se compone de dos cuadros, tendrá que detectar cuándo se encuentra a una posición del final de la pantalla.

Convierta esto en un juego de frontón, usando aquella rutina que comprobaba cuándo un objeto choca contra otro. Recordará que se usaban los comandos ATTR o SCREEN$ para conseguirlo. Esta vez usaremos ATTR, ya que el juego es en color:

```
  1 PAPER 7: CLS
  2 LET go=1: REM SQUASH
  3 CLS : PRINT AT 0,0;"VEZ=";go
  4 LET s=1: LET d=1
  5 LET g=(RND*8)+6: LET h=13
  6 LET x=14: LET y=15
  7 GO SUB 100
  8 PRINT AT x,y; INK 4;CHR$ 131+CHR$ 1 31
 10 IF INKEY$="5" THEN  GO TO 40
 11 IF g>14 THEN  GO TO 300
 13 PRINT AT g,h;"o"
 14 PAUSE 2
 15 GO SUB 200
 20 IF INKEY$="8" THEN  GO TO 50
 30 GO TO 10
 40 PRINT AT x,y;"  "
 41 LET y=y-1
 42 IF y<6 THEN  LET y=6
 43 PRINT AT x,y; INK 4;CHR$ 131+CHR$ 1 31
```

```
  44 GO TO 10
  50 PRINT AT x,y;"  "
  51 LET y=y+1
  52 IF y>24 THEN  LET y=24
  53 PRINT AT x,y; INK 4;CHR$ 131+CHR$ 1
31
  54 GO TO 10
  70 GO TO 20
 100 BORDER 1
 110 FOR a=5 TO 26
 120 PRINT  INK 2;AT 3,a;CHR$ 140
 130 NEXT a
 140 FOR a=4 TO 14
 150 PRINT  INK 5;AT a,5;CHR$ 138;AT a,2
6;CHR$ 133
 160 NEXT a
 170 RETURN
 200 PRINT  OVER 1;AT g,h;"o"
 201 PAUSE 2
 220 IF ATTR (g-d,h-s)=61 THEN  LET s=s*
-1
 230 IF ATTR (g-d,h-s)=58 THEN  LET d=d*
-1
 240 IF ATTR (g-d,h-s)=60 THEN  LET d=d*
-1
 245 LET g=g-d: LET h=h-s
 250 RETURN
 300 LET go=go+1
 305 PAUSE 50
 310 IF go>5 THEN  GO TO 400
 320 GO TO 3
 400 CLS : PRINT AT 10,12; PAPER 2; INK
6; FLASH 1;"FIN PARTIDA"
```

En alta resolución se usa POINT para saber si un punto de la pantalla ha sido ennegrecido con PLOT o no. Aquí tenemos como actúa:

```
10 PLOT 128,88
20 PRINT POINT (128,88)
```

Si ejecuta esto, verá un pequeño punto en el centro de la pantalla, y se imprimirá un 1 indicando que la posición 128,88 ha sido PLOTeada.

Si la posición no contiene punto, entonces el resultado de POINT es 0. Un simple ejemplo en el que se utilice esto puede ser:

```
5 LET a = 1
10 PLOT 0, 10: DRAW 255, 0
20 PLOT 0, 100: DRAW 255, 0
30 LET X = 120 : LET Y = 50
40 PLOT X,Y: PLOT OVER 1;X,Y
50 LET Y = Y + a
60 IF POINT (X,Y + A) = 1 THEN LET a = a* - 1
70 GO TO 40
```

Este programa dibuja dos líneas horizontales, y hace rebotar una pelota muy pequeña entre ellas. También puede usar POINT para saber si algo está en la pantalla o no. Por ejemplo, cuando ATTR no conviene y además se usan caracteres definidos (lo que impide la utilización de SCREEN$), entonces POINT es la única solución.

Obviamente, también se puede usar la rutina de las flechas para alta resolución, haciendo que las teclas 5 a 8 muevan un punto por la pantalla.

Un programa que dibuje (tipo tele-sketch) sería muy fácil de hacer usando esto. Lo único que hay que hacer es no borrar el último punto con OVER sino esperar a que se pulse una tecla (por ejemplo, la cero por DELETE) y entonces hacer que todos los PLOTs sean OVER hasta que se pulse otra tecla que anule esto (podría ser la P por PRINT).

ACELERANDO EL MOVIMIENTO

En muchos de los juegos en los que se utiliza el movimiento, puede ser difícil que el juego se ejecute a una velocidad aceptable. En el anterior programa del frontón, había sólo dos cosas que se movían, la pelota y la raqueta. Pero si hay más de dos cosas moviéndose en la pantalla al mismo tiempo, la velocidad puede reducirse mucho.

Vamos a considerar un simple juego de invasores, en el que por comodidad he usado la letra I para representar al invasor y un asterisco para representar el disparo.

```
10 FOR a = 0 TO 31
20 PRINT AT 5,a; "I";AT 5,a;"ʌ"
30 IF INKEY$ = "7" THEN GOSUB 200
40 NEXT a
50 GO TO 10
200 FOR b = 20 TO 5 STEP - 1
210 PRINT AT b,15;"*";AT b,15;"ʌ"
220 NEXT b
230 RETURN
```

Como puede observar, con esta rutina la letra I se para mientras se efectúa el disparo. Una manera un poco burda de arreglar esto consiste en colocar un PAUSE X entre cada movimiento de la I. X debe ser tal que la pausa producida sea más o menos igual que el tiempo que tarda en llegar el disparo. Esto se debe hacer probando cuál es el número más indicado. Una vez se ha detectado la pulsación de la tecla 7, se salta a X la subrutina del disparo haciendo que X tome el valor mínimo, de este modo parecerá que la I no se detiene al lanzarle el disparo.

Para conseguir mayor velocidad en este tipo de juegos sin necesidad de programarlos en código máquina (véase la 5.ª parte del libro) puede colocar los caracteres directamente en el archivo de imagen mediante POKE. Esto supone enviar los caracteres que se van a imprimir directamente a la pantalla sin usar PRINT que tarda bastante más tiempo.

```
10 LET C = 0
20 LET A = 16384 + C
30 FOR B = 0 TO 7
40 POKE A, PEEK (USR "a" + B)
50 LET A = A + 32*8
60 NEXT B
70 LET A = 16384 + C
80 FOR D = 0 TO 7
90 POKE A,0
100 LET A = A + 32*8
110 NEXT D
120 LET C = C + 1
130 IF C = 31 THEN LET C = 0
140 GO TO 20
```

Ejecutando esto verá cómo la letra A se escribe en la pantalla de arriba a abajo y luego es borrada. Esto es casi ideal, pero al añadir una rutina para el disparo, el movimiento de la A se hará más lento.

El modo en que se POKEa A en la pantalla es un poco complicado debido a la extraña disposición del archivo de imagen en el Spectrum. En el apéndice, encontrará una representación del mapa de memoria del Spectrum. Allí puede ver, que el archivo de imagen, ocupa desde la posición 16384 hasta 22527. La pantalla está organizada en grupos de ocho puntos (un byte), del siguiente modo: La pantalla contiene 24 líneas (contando las dos últimas), pues bien, el fichero está dividido en tres partes, cada una de las cuales, contiene la información de un grupo de ocho líneas. Hasta aquí muy bien, pero dentro de cada parte, las primeras 32 posiciones corresponden a la parte de arriba de la primera línea, pero las 32 siguientes no corresponden al segundo byte de los caracteres que forman la primera línea sino que contiene la información de los primeros bytes de los caracteres de la segunda línea; es decir, primero van las partes de arriba de cada línea, luego la segunda hilera de puntos, y así sucesivamente. Las otras dos partes se estructuran de igual forma. Para ver lo que quiero decir, lo mejor es POKEar el fichero entero y ver el orden que sigue el Spectrum para llenar la pantalla. Si usted ha utilizado SAVE y LOAD con SCREEN$ ya sabrá de que estoy hablando. Escriba:

```
10 FOR A = 16384 TO 22527
20 POKE A, 225
30 NEXT A
```

Como consecuencia de esta distribución del archivo de imagen, resulta que los ocho bytes que forman un carácter no están seguidos sino que se encuentran a 32*8 posiciones de memoria de distancia entre ellos. Esto es lo que se tiene en cuenta en las líneas 50 y 100 del programa anterior. La primera vez la pantalla se POKEa con el carácter del gráfico definido en la tecla A y luego con un cero para borrarlo. Como puede apreciar, dada esta particular estructura es más fácil mover un carácter horizontalmente que hacerlo verticalmente.

Otra cosa más, he escogido el carácter gráfico definible A porque se puede reemplazar fácilmente por una nave, un invasor, o cualquier otra cosa.

AUTOEJECUCIÓN AUTOMÁTICA

Si usted tiene un programa cuyo listado no quiere que sea visto por el jugador, probablemente habrá intentado distintos métodos para impedirlo, como el truco del listado invisible que se dio anteriormente. Hay otra cosa que se puede hacer con respecto a esto y que además añade profesionalidad a sus programas. Es hacer que sus programas se ejecuten automáticamente después de recuperarlos del cassette mediante LOAD. Para hacerlo, se usa de nuevo LINE de esta manera:

```
9999 SAVE "Programa" LINE 100
```

Cuando haya escrito el programa escriba GOTO 9999 y el programa empezará a grabarse él mismo y cuando lo recupere con LOAD empezará a ejecutarse automáticamente desde la línea 100. Esto útil porque no borra las variables. Por supuesto, que usted puede poner cualquier número después de LINE.

USANDO SCREEN$

Si ha gastado casi toda la memoria disponible, en un programa largo y ve que necesita más espacio, piense que puede usar SAVE y LOAD con SCREEN$ para almacenar en cinta pantallas llenas de información, que a lo mejor sólo se necesita para explicar el juego al principio, o que se puede indicar al jugador que haga LOAD cuando sea necesario. Esto proporcionará gran cantidad de espacio.

UN RELOJ

Muchos de los juegos que usted quiere escribir serían mejores si pudiera disponer de un reloj para que controlara que las cosas se hacen en un tiempo determinado, etc. El Spectrum dispone de un reloj interno, que ocupa tres bytes en la variable del sistema **FRAMES.**

FRAMES se encuentra en las posiciones de memoria 23672, 23673, y 23674. La menor de ellas, la 23673 empieza a cero, y se incrementa en 1 cada dos centésimas de segundo. Cuando alcanza el valor 255 vuelve a cero otra vez y pone un 1 en 23673. La siguiente vez, pone un 2 en 23673, y cuando ésta llega a 255 se pone a cero y coloca un 1 en 23674.

Esto tiene un pequeño problema y es que al leerse las posiciones secuencialmente puede ocurrir que 23674 dé cero, pero esté a punto de cambiar a 1 y cuando se lee 23673, entonces 23674 ya esté a 1 y por lo tanto 23673 sea 0, obteniéndose una respuesta muy baja. Esto se puede evitar leyendo dos veces y tomando la más alta. Esta expresión da el tiempo, en segundos, que ha transcurrido desde que estaban los tres a cero (en principio desde que conectó su Spectrum a la red), pero usted puede ponerlos a cero cuando quiera usando POKE:

[(65536*A + 256*B + C)/50]

En esta expresión, A es PEEK 23674, B es PEEK 23673, y C es PEEK 23672. Los minutos se obtienen dividiendo por 60, y las horas se obtienen también muy fácilmente. En cualquier momento puede colocar el reloj a cero o bien a una hora determinada. Para ponerlo a las 9.00 usted deberá calcular: 9*60*60*50 = 1620000. Dividiendo esto por 65536 y tomanto la parte entera (con INT) se obtiene el número que debe "POKEar" en 23674, el resto de la división anterior, se divide por 256 y obtiene lo que debe "POKEar" en 23673, y el resto de esta última división en 23672. Es decir:

1630000 = 65536*24 + 256*184 + 32

Y usted hará: POKE 23674, 24: POKE 23673, 184: POKE 23672, 32

CÓMO HACER QUE ESPERE

En muchos juegos, usted querrá que el jugador se tome su propio tiempo para tomar una decisión, o incluso para leer algo. Una manera muy común de detener el programa

en espera del jugador, es esperar a que éste pulse una tecla. Por ejemplo:

```
300 IF INKEY$ = ""THEN GOTO 300
```

Sin embargo, el Spectrum parece tener problemas al trabajar con INKEY$. Una prueba de esto es el ejemplo que sigue, en el que el programa no espera a que usted pulse una tecla para seguir:

```
10 IF INKEY$ = ""THEN GOTO 10
20 PRINT "AQUÍ EMPIEZA EL PROGRAMA"
```

Esto se debe a que cuando usted pulsa RUN y luego ENTER, este último, es tomado por INKEY$ con lo cual se deja de cumplir la condición. Esto se arregla fácilmente añadiendo:

```
5 PAUSE 2
```

Recuerde este hecho en cualquier programa que utilice INKEY$. Sin embargo, tal como apuntó el escritor del libro del ZX81, puede usarse una manera mucho más corta para esperar a que se pulse una tecla, que también funciona en el Spectrum:

```
10 PRINT "PULSE UNA TECLA PARA EMPEZAR"
20 PAUSE 4E 4
30 PRINT "EMPIEZA EL PROGRAMA"
```

Lo bueno de esto es que 4E4 suena casi como "para siempre" (for ever) y puede ser fácilmente recordado. Sin embargo, por casualidad descubrí un modo más corto de hacerlo y más fácil de recordar. Simplemente es:

```
20 PAUSE 0
```

En el ZX81, este comando no espera nada, pero en el Spectrum, se espera indefinidamente, hasta que se pulsa una tecla. La ventaja sobre el otro es que 4E4 no es mucho tiempo si la persona tiene que leer algo, entonces PAUSE 0 es la solución ideal.

Cuarta Parte

Aplicaciones serias y para la educación

CAPÍTULO NOVENO

Tomemos al Spectrum en serio

A pesar de su tamaño, el Spectrum puede ser útil en aplicaciones serias como negocios, educación y científicas. Con un poco de imaginación, puede sacarlo de las actividades de ocio que es para lo que se usa normalmente y convertirlo en una herramienta muy útil para el hogar o la oficina. Sin embargo, es quizás en un aula donde el Spectrum puede con su presencia causar gran efecto.

EDUCACIÓN

Como herramienta para la educación, el Spectrum es muy potente, ya que los niños, encuentran mucho más fácil aprender cuando lo hacen a su propio ritmo con un ordenador que tiene mucha paciencia. El uso inteligente de animación gráfica y color puede hacer del hecho de aprender una tarea interesante y absorbente. Es competencia y responsabilidad de los profesores y de las compañías de software el que todo esto se explote al máximo. En la escuela misma, el Spectrum puede servir para ayudar en la administración, guardar las notas de los exámenes, o incluso valorar lo que saben los niños cuando empiezan a leer.

Tomando esto último como ejemplo, no sería muy difícil hacer un programa en él; después de haberle introducido un texto, que analizara el número de frases que contiene y el número de palabras en cada frase. Para ello será muy útil la potencia que posee el Spectrum en cuanto a manipulación de cadenas se refiere. Simplemente habría que DIMensionar un gran conjunto en el que se introduciría el texto. Luego contar el número de frases mirando el número de puntos que contiene y mirando si les sigue una

mayúscula (ya que un punto puede pertenecer a una abreviación), puesto que raramente viene una mayúscula detrás de una abreviación. En todo caso este método dará la aproximación suficiente para el propósito del ejercicio. El número de palabras por frase se puede contar calculando el número de espacios que hay. Una vez hecho esto nos dará una media aproximada del número de palabras por frase que ha escrito el alumno. Este resultado puede interpretarse con tablas adecuadas.

NEGOCIOS

Con lo que ha aprendido en este libro, supongo que quedará claro que con el Spectrum es posible controlar STOCKS y gestionar archivos. Sin embargo, lo que ya dudo es que haya considerado la posibilidad de tener un procesador de textos con el Spectrum.

Quiero dejar claro desde el principio, que el Spectrum tiene en potencia un procesador de textos, aunque no podrá escribir muy deprisa debido al teclado. Sin embargo, es posible usar al Spectrum para informes, cartas, etc. El programa que sigue es un procesador de textos muy simple, que tal como está sólo admite una página de texto (una pantalla). Sin embargo, incorpora facilidades para insertar y borrar caracteres. Usted puede modificar el programa para que admita más texto, cambiando el conjunto de la línea 100 y detectando cuándo se llena una pantalla. Por ejemplo podría dimensionarlo de nuevo con A$ (N, 32*22) de este modo la primera dimensión le referirá a la página en que está, y la segunda será una pantalla llena de texto (22*32 caracteres). La justificación del texto, también es posible, simplemente usando rutinas para quitar o poner espacios. El método consiste en ver si cada bloque de 32 caracteres termina con un espacio, si no es así se busca el primer espacio hacia atrás y se cuenta a cuántos caracteres está del último de la línea. Este número obtenido, es el número de espacios que hay que distribuir por la línea entre las palabras. El texto sobrante se añade a la línea siguiente.

```
10 REM ***********
20 REM  MICROTEXT
30 REM ***********
```

```
  40 PAPER 6: BORDER 6: CLS
  45 REM INSTRUCCIONES
  Por primera vez se puede:
      -Entrar texto
      -Parar con s o S
    -Entrar en editor con e o E
  46 REM   MODO EDITOR
      -Flechas 5 y 8 para moverse
       -borrar caracter con d
       -insertarlo con i
       -para volver dar y
  50 LET EDIT=200
  60 LET INSERT=320
  70 LET DEL=390
  80 LET RET=110
  90 LET A=1
 100 DIM T$(32*21)
 110 INPUT "TU SENTENCIA?(S=STOP,P=ESCRI
BE)"; LINE S$
 115 PRINT AT 0,0;T$
 120 PRINT AT INT (A/32),A-32*INT (A/32)
;
 130 IF S$="e" OR s$="E" THEN  GO TO EDI
T
 140 IF S$="p" OR s$="P" THEN  LPRINT T$
 150 IF S$="s" OR s$="S" THEN  STOP
 160 PRINT S$;
 170 LET T$(A TO A+LEN S$)=S$
 180 LET A=A+LEN S$+1
 190 GO TO RET
 200 REM ***EDITAR***
 210 LET W=1
 220 CLS
 230 PRINT AT 0,0;T$( TO W); FLASH 1;"*"
; FLASH 0;T$(W+1 TO );
 240 IF INKEY$="8" THEN  LET W=W+1
 250 IF INKEY$="5" THEN  LET W=W-1
 260 IF INKEY$="d" THEN  GO SUB DEL
 270 IF INKEY$="i" THEN  GO SUB INSERT
 280 IF INKEY$="y" THEN  GO TO RET
 290 IF W<1 THEN  LET W=1
 300 IF W>(32*21-1) THEN  LET W=(32*21)-
1
 310 GO TO 230
```

```
320 REM ***INSERCION***
330 INPUT "QUE INSERTAS? "; LINE I$
340 LET T$(W+LEN I$+1 TO )=T$(W TO )
350 LET T$(W+2 TO W+2+LEN I$)=I$
360 PRINT AT INT ((A+3)/32),A+3-32*INT
((A+3)/32);
370 LET A=A+6
380 RETURN
390 REM ***BORRADO***
400 LET T$(W TO )=T$(W+1 TO )
410 LET A=A-1
420 RETURN
```

Una extensión del programa anterior consistiría en permitir sacar una carta por la pantalla y que se pudiera añadir texto o figuras donde sea necesario. Hay dos maneras de hacer esto, y ambas usan SCREEN$.

El primer método consiste en poner la carta en la pantalla y luego asignar todo el texto a una cadena usando SCREEN$.

```
10 DIM S$ (32*22)
20 LET X = 1
30 FOR A = 0 TO 21
40 FOR B = 0 TO 31
50 LET S$(X) = SCREEN$ (A,B)
60 LET X = X+1
70 NEXT B:NEXT A
```

Cuando usted quiera usar esta carta o documento sólo tiene que ponerlo en pantalla escribiendo:

```
PRINT AT 0,0;S$
```

El programa debería estar escrito de manera que preguntara lo que hay que añadir a la carta, y luego lo imprimiera en la pantalla en el lugar correcto. La pantalla completa puede copiarse en la impresora con COPY o registrarla en cassette mediante SAVE SCREEN$.

El otro método, consiste precisamente en usar SAVE ""SCREEN$. Una vez tiene la pantalla con el texto se

graba en cassette con SAVE SCREEN$ y entonces se escribe un programa tal que, una vez la pantalla se ha recuperado, le permita mediante INPUT añadir lo que sea necesario al texto. Una vez completado puede imprimirlo, o bien puede guardarlo en cassette de nuevo.

Quizás uno de los programas más útiles para un hombre de negocios sea un buen sistema de archivo. Aquí hay uno que puede manipular hasta tres columnas de información que son definidas por usted mismo. Lo que puede almacenar en el fichero depende de qué Spectrum tenga usted. Este programa ha sido escrito para el Spectrum de 48K y tiene unos conjuntos en la línea 5 que permiten introducir 200 campos más o menos. Con el Spectrum de 16K estos conjuntos tendrían que ser de 300 × 10 y permitirían introducir unos 50 campos aproximadamente. Por supuesto que cuando lleguen los microdrives podrá arreglar este programa para que trabaje con ellos, y podrá disponer de muchísimos campos más.

```
    2 REM ***********************
    3 REM * sistema de ficheros *
    4 REM ***********************
    5 REM Graham J Scott.
    6 DIM a$(1200,10): DIM b$(1200,10): D
IM c$(1200,10): LET a=1: LET b=0: LET e$
=""
   10 PAPER 1: BORDER 1: INK 5
   14 PRINT AT 10,5; INVERSE 1;"SISTEMA D
E FICHEROS"
   15 PRINT '''"PRESIONE UNA TECLA PARA CO
NTINUAR": PAUSE 0
   20 CLS : LET d$="": PRINT TAB 13;"MENU
"
   22 PRINT TAB 13;""
   25 PRINT ''"  1 para crear fichero"
   30 PRINT '"  2 para listar fichero"
   40 PRINT '"  3 para guardar fichero"
   60 PRINT '"  4 para cargar fichero"
   75 PRINT '"  5 para buscar"
   77 PRINT '"  6 para annadir fichero"
   78 PRINT '"  7 para imprimir"
   79 PRINT '"  8 para terminar"
   80 PRINT ''TAB 2; FLASH 1;"Entre la op
cion deseada"
```

```
  85 LET d$=INKEY$
  90 IF d$="" THEN  GO TO 85
  93 IF d$>"8" OR d$<"1" THEN  GO TO 85
  95 GO TO VAL d$*100
 100 REM *****Crear fichero*****
 101 CLS : PRINT TAB 5; INVERSE 1;"CREAC
ION FICHERO"
 102 LET a=0
 103 PRINT AT 10,3; INVERSE 1;"stop"; IN
VERSE 0;" para finalizar"
 104 PRINT AT 12,3; INVERSE 1;"del"; INV
ERSE 0;" para borrar ultimo dato"
 105 PRINT AT 15,10;"PRESIONE UNA TECLA"
: PAUSE 0
 106 CLS : INPUT "Entre los titulos de l
as 3 columnas";x$;y$;z$
 108 GO SUB 1020
 110 LET a=a+1
 115 LET b=b+1: IF b=22 THEN  LET a=a-1:
 LET b=0: GO TO 107
 120 INPUT  LINE a$(a)
 124 IF a$(a)="stop        " THEN  LET a$(
a)="            ": GO TO 160
 125 IF a$(a)="del         " THEN  LET a=a
-1: LET b=b-1: PRINT AT b,22;: GO TO 140
 127 PRINT a$(a);
 130 INPUT  LINE b$(a)
 132 IF b$(a)="del         " THEN  PRINT A
T b,0;: GO TO 120
 135 PRINT TAB 11;b$(a);
 140 INPUT  LINE c$(a)
 142 IF c$(a)="del         " THEN  PRINT A
T b,11;: GO TO 130
 145 PRINT TAB 22;c$(a);
 150 GO TO 110
 160 CLS : PRINT AT 10,9; PAPER 6; INK 2
; FLASH 1;"O R D E N A N D O "
 170 FOR z=1 TO a: FOR t=1 TO a+2: LET e
$=a$(t): LET f$=b$(t): LET g$=c$(t): IF
a$(t+1)>=a$(t) THEN  GO TO 190
 180 GO TO 195
 190 LET a$(t)=a$(t+1): LET b$(t)=b$(t+1
): LET c$(t)=c$(t+1): LET a$(t+1)=e$: LE
T b$(t+1)=f$: LET c$(t+1)=g$
```

```
 195 NEXT t: NEXT z: GO TO 20
 200 REM *****Listar fichero****
 202 IF a$(1)="          " THEN  GO TO 2
0
 205 LET d=a+1
 207 GO SUB 1020
 210 LET e=0
 220 LET d=d-1
 222 LET e=e+1
 224 IF d=0 THEN  GO SUB 1000: GO TO 20
 225 IF e=20 THEN  LET e=0: GO SUB 1000:
 GO SUB 1020
 230 PRINT a$(d);TAB 11;b$(d);TAB 22;c$(
d)
 240 GO TO 220
 300 REM ****Guardar fichero****
 310 CLS : PRINT ''"Como se llama el fic
hero?": INPUT  LINE f$
 320 SAVE f$ LINE 420
 330 GO SUB 1000
 340 GO TO 20
 400 REM ****Grabar fichero****
 405 CLS : PRINT ''"Como se llama el fic
hero?": INPUT  LINE f$
 410 LOAD f$
 420 GO TO 20
 500 CLS : PRINT ''"Columna a buscar?"
 502 LET e=0
 505 PRINT ''"     1) ";x$: PRINT '"     2
) ";y$: PRINT '"     3) ";z$
 510 LET d$=INKEY$: IF d$="" THEN  GO TO
 510
 515 IF d$>"3" OR d$<"1" THEN  GO TO 510
 516 PRINT ''"Dato a buscar?": INPUT  LI
NE a$(1200)
 518 GO SUB 1020
 520 GO TO (VAL d$*20)+510
 530 FOR i=a TO 1 STEP -1: IF a$(i)=a$(1
200) THEN  LET e=e+1: GO SUB 590
 540 NEXT i: GO SUB 1000: GO TO 20
 550 FOR i=a TO 1 STEP -1: IF b$(i)=a$(1
200) THEN  LET e=e+1: GO SUB 590
 560 NEXT i: GO SUB 1000: GO TO 20
 570 FOR i=a TO 1 STEP -1: IF c$(i)=a$(1
```

```
200) THEN  LET e=e+1: GO SUB 590
 580 NEXT i: GO SUB 1000: GO TO 20
 590 IF e=21 THEN  LET e=0: GO SUB 1000:
 GO SUB 1020
 595 PRINT a$(i);TAB 11;b$(i);TAB 22;c$(
i)
 596 RETURN
 600 GO SUB 1020: LET b=0: GO TO 115
 700 CLS : LET d=a+1
 710 LPRINT x$;TAB 11;y$;TAB 22;z$
 720 LPRINT "                           "
 730 LET d=d-1: IF d=0 THEN  GO TO 20
 740 LPRINT a$(d);TAB 11;b$(d);TAB 22;c$
(d): GO TO 730
 800 CLS : STOP
1000 PRINT AT 21,1; FLASH 1; PAPER 6; IN
K 2;"Pulse una tecla para continuar"
1010 PAUSE 0: RETURN
1020 CLS : PRINT x$;TAB 11;y$;TAB 22;z$
1030 PRINT AT 0,0; OVER 1;"-------------
-------------------";: OVER 0
1040 RETURN
```

APLICACIONES CIENTÍFICAS

Debido a su gran capacidad matemática el Spectrum tiene numerosas aplicaciones científicas. Posee muchas de las funciones que uno espera encontrar en las calculadoras más completas. Estas funciones incluyen: SIN, COS, TAN, ARCSIN, ARCOS, ARCTAN. Además posee raíces cuadradas y exponenciales así como funciones tan útiles como ABS y SGN, y la capacidad de definir complejas ecuaciones matemáticas con DEF FN.

Aquí tiene un ejemplo de un uso típicamente científico del Spectrum. Este programa calcula el logaritmo en base 10 de cualquier número.

```
10 REM LOGARITMO EN BASE 10
20 INPUT "¿NUMERO?", N
30 LET X = (LN N)*.4343
40 PRINT "logaritmo en base 10 de"; N; "ES"; X
```

Para los estudiantes de ciencias sociales, aquí hay un programa estándar de estadística:

```
   5 REM t-TEST de STUDENT
  10 LET T1=0: LET R1=0: LET T=0: LET R=
0
  20 INPUT "CUANTOS SUJETOS HAY EN EL GR
UPO 1?",N1
  30 INPUT "CUANTOS SUJETOS HAY EN EL GR
UPO 2?",N2
  35 DIM X(N1): DIM Y(N2)
  40 PRINT AT 0,0;"DATOS PARA EL GRUPO 1
:"
  50 FOR A=1 TO N1
  60 INPUT X(A)
  65 PRINT X(A);",";
  70 NEXT A
  75 CLS
  80 PRINT AT 0,0;"DATOS PARA EL SEGUNDO
 GRUPO:"
  90 FOR A=1 TO N2
 100 INPUT Y(A)
 105 PRINT Y(A);",";
 110 NEXT A
 115 REM SUMA DE CUADRADOS +
 CUADRADO  DE SUMA
 120 FOR A=1 TO N1
 130 LET T=T+X(A)↑2
 140 LET T1=T1+X(A)
 150 NEXT A
 160 FOR A=1 TO N2
 170 LET R=R+Y(A)↑2
 180 LET R1=R1+Y(A)
 190 NEXT A
 191 CLS : PRINT "RESULTADO"'
 195 LET MEAN1=T1/N1: LET MEAN2=R1/N2
 200 LET R1=R1↑2: LET T1=T1+2
 210 LET SS1=T-(T1/N1)
 211 PRINT "SS1=";SS1,
 220 LET SS2=R-(R1/N2)
 221 PRINT "SS2=";SS2
 230 LET W=(MEAN1-MEAN2)/(SQR (((SS1+SS2
)/((N1-1)+(N2-1))*((N1↑-1)+(N2↑-1)))))
 240 PRINT "MEDIA 1=";MEAN1,"MEDIA 2=";M
EAN2
 250 PRINT "t=";W
```

Por último, si usted posee un puerto de E/S para el Spectrum, entonces le será bastante fácil programarlo para controlar un equipo de laboratorio, tomar resultados de experimentos, etc.

Quinta Parte

Usar el Spectrum hasta el límite

CAPÍTULO DÉCIMO

¡Refuerce sus programas!

MEJORAR LA PROGRAMACIÓN

Una de las cosas más esenciales que se necesitan para programar bien es pensar de un modo lógico y estructurado en el problema que se nos presenta. También debe procurar hacer los programas inteligibles para los demás (a menos que tenga una razón específica para no hacerlo), porque así también serán claros para usted cuando vuelva a ellos después de un tiempo.

Estructurar el programa claramente le ayudará mucho. Si la capacidad de memoria se lo permite, no escatime sentencias REM para clarificar el programa, dividiéndolo en partes e indicando qué trabajo efectúan cada una de ellas. Para hacerlo más fácil de leer puede usar REMs en blanco alrededor de los verdaderos REMs. Si el espacio en memoria se lo permite puede incluir líneas en blanco que también ayudarán a una mejor lectura del programa. El Spectrum acepta líneas en blanco escribiendo el número de línea seguido al menos de un espacio. Esto coloca agujeros en el listado haciéndolo más fácil de leer pero sin gastar demasiada memoria:

```
10 LET A = 4,87
20
30 A < B THEN PRINT "HA PERDIDO"
```

En el ejemplo de arriba, la línea 20, que está vacía, nos marca la diferencia entre el programa y la rutina que comprueba si el jugador ha perdido.

El Spectrum también tiene la particularidad de poder colorear el listado sin que esto tenga efecto cuando el

programa es ejecutado. Recuerde que esto se hace poniendo el cursor en modo E y pulsando la tecla del color correspondiente.

Tenga cuidado cuando quiera listar programas por la impresora pues ésta puede producir problemas al intentar leer estos caracteres de control.

Si el programa contiene un cierto número de rutinas que se deben ejecutar una sola vez entonces hágalo al principio del programa. Coloque las subrutinas al final del programa y acceda a ellas desde las primeras líneas poniendo un REM antes del GOSUB que indique lo que va a hacer:

```
10 REM CREACIÓN DE UN CARÁCTER DE
INVASOR
20 GOSUB 9000
```

El Spectrum también permite dar nombres completos a las variables numéricas, y este hecho se puede usar para crear un BASIC semiestructurado en su ordenador. Si usted asigna a una variable llamada PRUEBA un número que no es más que el número de línea donde ocurre la PRUEBA, entonces podrá decir GOTO PRUEBA o GOSUB PRUEBA en cualquier parte del programa. Esto proporciona ventajas similares a las "procedures" de otros ordenadores (que permiten definir un proceso y llamarlo desde cualquier punto del programa). Observe que asignando números de línea a las variables, hará mucho más fácil la tarea de mirar un área específica de su programa. Por ejemplo, si cada subrutina de su programa tiene un nombre y este nombre se usa como variable conteniendo el número de línea donde empieza la subrutina, entonces simplemente haciendo LIST nombre tendrá la rutina en la pantalla. Por ejemplo:

```
10 LET PUNTUACIÓN = 1000
1000 REM RUTINA DEL CÁLCULO DE LA
PUNTUACIÓN
```

Así, LIST PUNTUACIÓN, listará de la línea 1000 en adelante. Esto también tiene la ventaja de que si usted usa una rutina de renumeración que deja los mismos números en los GOTO y GOSUB, solamente necesitará cambiar

estas variables al principio del programa en lugar de tener que buscar todos los GOSUBs y GOTOs.

También puede hacer que todos los procedimientos similares aparezcan en el listado del mismo color, las líneas de datos, de otro color, etc.

Las variables que use con mucha frecuencia, defínalas al principio del programa. Esto lo digo porque cuando el Spectrum encuentra una variable la busca empezando desde el principio del programa hasta que la encuentra. El programa irá por lo tanto más rápido, si la variable se encuentra enseguida.

Piénselo un poco antes de hacer la parte visual de su programa. Muchos programas excelentes pierden mucho debido a una pobre presentación gráfica, textos mal centrados, colores que no ligan, etc. Si durante el programa el usuario tiene que leer un texto, use PAUSE 0 para que pueda hacerlo durante el rato que desee, hasta pulsar una tecla.

Pruebe exhaustivamente sus programas, pero no sólo introduciendo lo que ya cabe esperar sino todo lo que se le ocurra. Controle las entradas de datos mediante las rutinas que ya se explicaron en su momento. No use preguntas ambiguas que pueden no significar lo mismo para usted que para los demás.

RENUMERACIÓN DE LÍNEAS

Una vez terminado un programa, rara vez quedan las líneas a la misma distancia (los números, claro) sino que hay áreas en las que se han tenido que añadir líneas y sus números sólo difieren en una o dos unidades, mientras que en otros lugares se han borrado, con el consiguiente desequilibrio en la numeración. Lo peor de todo esto no es el desorden, sino que a veces puede suceder que no disponga de más números para colocar en un área del programa que necesita ser desarrollada. La solución está en una rutina de renumeración, que lo que hace es renumerar el programa empezando a partir de un número de línea dado y con un paso marcado. Le sugiero que grabe este programa en cinta con estos mismos números de líneas que contiene, y así cada vez que termine un programa puede recuperarlo con MERGE:

```
9990 RENUMBER
9991 LET X = PEEK 23635 + 256*PEEK 23636
9992 LET FLN = 10: Primer n.º de línea
9993 LET LNS = 10: Paso entre líneas
9994 IF PEEK (X + 1) + 256*PEEK X = 9990 THEN
     STOP
9995 POKE X, INT(FLN/256)
9996 POKE X + 1,FLN – 256*INT (FLN/256)
9997 LET FLN = FLN + LNS
9998 LET X = X + 4 + PEEK(X + 2) + 256*PEEK
     (X + 3)
9999 GOTO 9994
```

CLEAR Y RESTORE

Si tiene un programa que ya contiene sus propias variables y no quiere que se mezclen con las de un programa anterior (si ha usado MERGE), o con las de él mismo si es la segunda vez que lo ejecuta, entonces empiécelo con esta línea:

```
1 CLEAR: RESTORE
```

VELOCIDAD

Para conseguir que sus programas se ejecuten a la máxima velocidad, intente usar el menor número de GOTOs posible y reserve los GOSUBs para las primeras líneas. El uso de números directamente en lugar de variables también aumentará la velocidad de ejecución. Si tiene una lista de condiciones de modo que si una es falsa, las otras no necesitan ser comprobadas; reemplácelo:

```
10 IF A < > 1 AND A < > 2 AND A < > 3 AND . . .
```

por:

```
10 IF A < > 1 THEN IF A < > 2
   THEN IF A < > 3 THEN . . .
```

La segunda versión irá a la línea siguiente en cuanto no se cumpla una condición, mientras que en la primera se evaluarán todas.

Cuando tenga que entrar largas listas de datos, le recomiendo que antes haga el siguiente POKE para cambiar el "clic" del teclado con objeto de que se oiga más. De este modo reducirá bastante los errores:

POKE 23609, 100

También se puede cambiar el tiempo que tarda una tecla en repetirse, y el tiempo que pasa entre dos repeticiones. Esto ayudará cuando tenga que mover el cursor por largas líneas de datos.

TIEMPO PREVIO A LA REPETICIÓN:
POKE 23561,x (pruebe con x = 10)

TIEMPO ENTRE REPETICIONES:
POKE 23562,x (x = es demasiado rápido,
x = 3 es más manejable)

CAPÍTULO UNDÉCIMO

Código máquina para el Spectrum

PROGRAMANDO EN CÓDIGO MÁQUINA

He mencionado el código máquina una y otra vez, pero ¿qué es exactamente? Recordará que al principio del libro, le mencioné que la parte principal del ordenador (la CPU) sólo entendía variaciones entre alto y bajo voltaje que se podían pensar como dos estados o como unos y ceros. Para comunicarse con la CPU se desarrolló un lenguaje parecido al inglés que se llama BASIC. El programa que reside en la ROM, convierte este lenguaje parecido al inglés en el lenguaje de unos y ceros que es el que entiende la CPU. Como es natural transcurre bastante tiempo mientras el programa que está en la ROM ordena las instrucciones BASIC y las traduce a lenguaje de unos y ceros.

Sin embargo, hay una alternativa que consiste en usar un lenguaje, que está muy cerca del de la máquina y que se llama código máquina. Que no es más que darle a cada uno de los números que componen una instrucción en el lenguaje de unos y ceros, un nombre para que sea más fácil de recordar. A estos nombres se les llama mnemónicos. Usando estos mnemónicos, el lenguaje máquina no está tan lejos del BASIC. Por ejemplo aquí tenemos una instrucción en lenguaje máquina con su equivalente en BASIC:

MNEMÓNICO	LITERAL	BASIC
1d A,34	asigna a A 34	LET A = 34

Como puede ver es cómo escribir abreviaciones en lugar de palabras enteras.

Pero tiene restricciones. En código máquina no se dispone de los lujosos bucles FOR NEXT ni de comandos tan comunes como PRINT. Veamos este ejemplo:

```
200 LET 4 = 23
210 FOR X = 1 TO 100
220 IF X > Y THEN GOTO 20
230 NEXT X
```

Ahora escribiremos el mismo programa sin usar el bucle FOR NEXT. Sería algo así:

```
200 LET X = 1
210 LET Y = 23
220 IF X>Y THEN GOTO 20
230 LET X = X + 1
240 IF X = 100 THEN STOP
250 GOTO 220
```

De este modo el programa es un poco más laborioso. Ahora, para que se parezca más al lenguaje máquina sustituiremos los IF THEN por otro tipo de comparaciones, ya que en lenguaje máquina sólo se puede comparar si una cosa es 0 o no lo es:

```
200 LET X = 0
310 LET B = 23
220 LET A = B
230 LET A = A - 28
240 STOP IF ZERO
250 LET X = X + 1
260 LET A = X
270 LET A = A - 100
280 STOP IF ZERO
290 GOTO 210
```

Aunque el lenguaje usado es distinto, y en lenguaje máquina no existen los números de línea, esta rutina se parece mucho a su equivalente en lenguaje máquina. Como puede ver requiere un poco más de concentración. Ahora veamos cómo sería exactamente nuestro programa en código máquina:

999	LD B,100	—Carga el registro B con 100 (LET B = 100)
1000	LD A,B	—Carga el registro A con B (LET A = B)
1001	CP 23	—Compara con 23 (LET A = A — 23)
1002	RET Z	RETURN si el resultado es 0
1003	DEC B	Decrementa B (LET B = B — 1)
1004	LD A,B	—Carga el registro A con B (LET A = B)
1005	CP 0	—Compara con 0 (LET A = A — 0)
1006	RET Z	—RETURN si el resultado es 0
1007	JP 1000	—Salta a la línea 1000 (GOTO 1000)

Primero una observación: en lenguaje máquina no se utilizan los números de línea, sino que los saltos a una instrucción deben ser enviados a la dirección de memoria en que ésta se encuentra. Aquí he puesto al lado de cada instrucción un número que representa la posición de memoria en que se encuentra, aunque no es un ejemplo muy real ya que las posiciones 999-1007 forman parte de la ROM, y en ellas no es posible colocar ninguna instrucción en código máquina.

Las líneas de las direcciones 1002 y 1006 tienen un RET Z que puede parecerle un poco complicado. Esto simplemente significa RETURN, casi lo mismo que hace un RETURN al final de una subrutina en BASIC. Sólo que aquí el programa posiblemente es llamado desde otro que está en BASIC (y llamar es similar a GOSUB), así que lo que hace esta línea es ver si el último resultado calculado es cero y si esto sucede, vuelve al programa en BASIC para ver lo que sigue.

Se estará preguntando por qué he hecho B igual a 100 al principio y le resto uno cada vez, mirando si es cero, en lugar de incrementarlo para hacer que el programa fuera paralelo con su homólogo en BASIC. Pues bien, lo he hecho para demostrarle que en algunas ocasiones, el lenguaje máquina es más potente que su equivalente en BASIC. A partir de ahora no podrá decir que para hacer la misma operación el lenguaje máquina necesita de tres instrucciones más que su equivalente en BASIC porque las líneas que van de la 1003 a la 1006 ambas inclusive, se pueden reemplazar por la instrucción: DJNZ e

Esta instrucción significa: "Decrementa B y salta 'e' posiciones hacia atrás hasta que el resultado sea 0". En este caso 'e' debe ser un número que haga saltar el programa a la línea 1000.

En esta introducción no he profundizado en absoluto en cuanto a la escritura de código máquina se refiere, pero espero haber dicho lo suficiente para que se sienta tentado de leer algún libro introductorio. De todas maneras voy a comentar otros aspectos. En el ejemplo anterior, he usado números decimales, pero el código máquina se escribe con números **hexadecimales.** La aritmética **hexadecimal** (normalmente llamada hex) es una manera de contar de 16 en 16, en vez de en decenas como es normal, o de 2 en 2 como ocurre en binario. Esto significa que las cifras de un número no tienen estos valores:

$$10^7 \quad 10^6 \quad 10^5 \quad 10^4 \quad 10^3 \quad 10^2 \quad 10^1 \quad 10^0$$

sino estos:

$$16^7 \quad 16^6 \quad 16^5 \quad 16^4 \quad 16^3 \quad 16^2 \quad 16^1 \quad 16^0$$

Así, con la primera columna de la derecha se puede contar hasta 15, con la segunda hasta 15, 16, etc. Contar hasta 15 en cada columna es difícil si sólo disponemos de 10 dígitos (del 0 al 9). Por eso en hex, se usan las letras de la A a la F para representar los números del 10 al 15. Así, 0A es 10, 0F es 15, 10 es 16, FF es 255, etc.

Más tarde veremos un programa para invertir los colores de la tinta y del papel en la pantalla.

DÓNDE ALMACENAR EL CÓDIGO MÁQUINA

El programa en código máquina tiene que almacenarse en algún lugar de la memoria. Recuerde que no contiene números de línea, con lo que es muy importante conocer la dirección de cada byte del programa. Si usted ya ha visto algo de lenguaje máquina, quizá con el ZX81, entonces habrá oído decir que es una buena idea colocar el programa en una sentencia REM. Lo que con esto se quiere decir es que primero usted llena 'n' bytes del área de programa (donde 'n' es el número de bytes que ocupa su programa en código máquina) colocando cualquier cosa

detrás de un REM y que estos bytes pueden ser sustituidos luego por instrucciones en código máquina sin afectar para nada al programa en BASIC. En el Spectrum no es necesario colocar el programa en una sentencia REM sino que lo verdaderamente importante es conocer exactamente en qué dirección de memoria empieza el programa. En otros ordenadores, esta dirección es fija (el ZX81, por ejemplo), sin embargo, en el Spectrum puede variar esta dirección según si están conectados dispositivos externos como los microdrives, etc. (vea el mapa de memoria que está en el apéndice). Por suerte, si usted usa REMs para registrar su programa siempre podrá determinar la dirección del comienzo usando la información que nos dan las variables del sistema (vea pág. 174 del manual). Hay dos posiciones, la 23635 y la 23636 que contienen el número que indica la dirección de memoria en que empieza el área de programa. Para hacerlo, use esta fórmula:

```
PRINT PEEK 23635 + 256* PEEK 23636
```

Un sitio aún mejor para almacenar el código máquina es hacerlo por encima de **RAMTOP,** y esto se puede hacer fácilmente usando CLEAR. Por ejemplo CLEAR 30000 limpia la memoria a partir de la posición 30000 y coloca a RAMTOP en dicha posición, con lo que de aquí en adelante todas estas direcciones quedan protegidas de ser borradas con NEW. RAMTOP indica la posición de memoria más alta que puede contener programas BASIC. Esta posición se puede variar para permitir más sitio donde registrar código máquina o datos simplemente. Para editar más fácilmente un programa en código máquina puede ponerlo en líneas DATA. Así, una buena rutina para hacerlo sería:

```
 5 CLEAR 30000
10 RESTORE 200
20 FOR X = 30001 TO 30001 + N (donde  W  es el
30 READ A                      número de bytes)
40 POKE X,A
50 NEXT X
200 DATA . . . . . .
```

Una vez el programa ya se ha colocado por encima de RAMTOP, puede grabarlo en cinta para recuperarlo cuan-

do lo desee, incluso si hay otro programa en memoria. Para hacerlo utilice los comandos LOAD y SAVE junto con CODE:

SAVE "nombre" CODE 30001,N

En este caso, la primera dirección que se grabará será la 30001 y luego los N bytes siguientes. Para recuperar el programa se hace lo mismo con LOAD.

Para los que deseen trabajar en hex, y almacenar el código máquina en sentencias REM aquí tienen un programa para hacerlo (use letras minúsculas para entrar los códigos):

```
   2 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   3 BORDER 1: INK 7: PAPER 1: CLS
   4 PRINT  INVERSE 1;" Hex-dec converti
dor   o  hex-cargador"''; INVERSE 1;"Pul
se 'c' para conversion, 'l' para cargar.
      Si pulsa "; FLASH 1;"STOP";
   5 PRINT ; FLASH 0; INVERSE 1;" se det
endra el programa"
   7 LET a=5+(PEEK 23635+256*PEEK 23636)
   8 LET t=0
   9 INPUT q$
  10 IF q$="l" THEN  LET t=1
  20 IF q$="c" THEN  GO TO 50
  25 CLS
  30 IF q$="l" THEN  PRINT  INVERSE 1;"h
ave you set up ..."
  40 INPUT o$: IF o$="s" THEN  STOP
  50 CLS
  60 PRINT "enter m..."
  98 LET x=2
  99 LET y=0
 100 INPUT h$
 105 IF CODE h$>102 OR CODE h$(2)>102 TH
EN  GO TO 240
 110 LET c=CODE h$
 120 IF c>90 THEN  GO TO 220
 130 LET c=c-48
 140 LET i=CODE h$(2)
 150 IF i>90 THEN  GO TO 220
 160 LET i=i-48
```

```
170 LET d=16*c+i
175 PRINT AT 20,0;"
180 PRINT AT x,y;d
181 IF t=1 THEN  GO TO 300
185 LET y=y+4
186 IF y=32 THEN  LET x=x+1
187 IF y=32 THEN  LET y=0
190 GO TO 100
200 LET c=c-87
210 GO TO 140
220 LET i=i-87
230 GO TO 170
240 PRINT AT 20,0;"Mal Hex;vuelva a ent
rar"
250 GO TO 100
300 POKE a,d: LET a=a+1: GO TO 185
```

INVERSIÓN DE LA PANTALLA

Tal como prometí, aquí hay un programa en código máquina que intercambia los colores del papel y de la tinta. Pero no únicamente cambia los colores (que se puede hacer directamente con POKEs en el fichero de atributos) sino que transforma cada carácter en su inverso. La ventaja de esto es que se puede imprimir en la impresora. He puesto esta rutina en la posición 30000, por lo tanto recuerde que antes debe hacer CLEAR 29999

DIRECCIÓN	CÓDIGO	MNEMÓNICO
7530	2AFF3F	LD HL (3FFF)
7533	23	INC HL
7534	7C	LD A,H
7535	FE58	CP 58
7537	C8	RET Z
7538	7E	LD A,(HL)
7539	2F	CPL
753A	77	LD (HL),A
753B	18F6	JR 7533

Esta rutina puede llamarse desde el BASIC escribiendo RAND USR 30000. El usar RAND USR en lugar de PRINT USR tiene la ventaja de que se ejecuta el programa y cuando termina no imprime nada en la pantalla. La forma

en que actúa el programa es muy simple. El registro de 2 bytes llamado HL es cargado con el valor de la dirección en donde empieza el archivo de imagen (es decir, en qué lugar se encuentra la pantalla en memoria), menos uno (ya que enseguida se incrementa). El registro A se carga con el contenido de H, y en la posición 7535 se compueba que no supere el valor 58, ya que si esto ocurre quiere decir que hemos llegado al final del archivo de imagen y se detiene el programa en 7537. Si no se ha llegado al final del archivo de imagen, entonces el contenido de HL está apuntando a una posición de la pantalla cuyo carácter queremos invertir. Esto se hace cargando en A el byte contenido en la dirección indicada por HL (en el mnemónico este hecho se representa colocando HL entre paréntesis) y luego tomando el CoMPLemento de A. El complemento de un código de carácter es justamente su imagen inversa que es lo que queremos. Ahora lo colocamos de nuevo en la pantalla efectuando la operación inversa en 735A. Una vez hecho esto, la próxima instrucción le dice al programa que salte 10 bytes hacia atrás donde HL se incrementa de nuevo y se repite todo el proceso para una nueva zona de la pantalla.

Si como espero aún no ha tenido bastante, aquí hay otra rutina que cambia los colores pero sin cambiar para nada el texto. Esto se puede hacer también usando POKE pero es mucho más lento:

	Decimal Bytes	Mnemónico	Comentario
	6,235	LD B,235	
	22,40	LD, D,40	Código Ink/Paper
	33,0,88	LD HL,22528	Comienzo del archivo de atributos.
dirección 32528	114	LD (HL), D	
	35	INC HL	
	114	LD(HL), D	
	35	INC HL	
	114	LD(HL, D	
	35	INC HL	
	5	DEC B	¿Es el final del archivo?
	194,16,127	JPNZ,32528	
	122	LD A,D	

```
50,141,92  LD(23693), A Hace permanentes
                        los colores
201        RET
```

El código del color y la tinta puede cambiarlo, se encuentra en la segunda línea y yo he puesto 40, pero usted puede poner cualquier otro número. Recuerde que este número es el de ATTR y se calcula como: 8 multiplicado por el código del papel más el código de la tinta. Luego deberá pasarlo a hexadecimal.

CÓMO OBTENER EL MÁXIMO DE MEMORIA

Si usted necesita más sitio en memoria para programas o datos, aquí tiene algunos trucos.

Puede desplazar el comienzo del área de los caracteres definidos por el usuario para obtener más sitio para su programa en BASIC. Esto depende, claro, del número de gráficos definibles que use en este programa. Pero si no usa ninguno, conseguirá 21*8 bytes más. Este lugar, también es bueno para almacenar código máquina. La dirección es siempre conocida ya que tanto en el Spectrum de 16K como en el de 48K, USR "a" nos da el comienzo de esta área. Así un programa para entrar código máquina utilizando esto sería:

```
10 FOR A = USR "a" TO USR "h"
20 READ X: POKE a,X
30 NEXT a
40 DATA . . . . . . . . . . . .
```

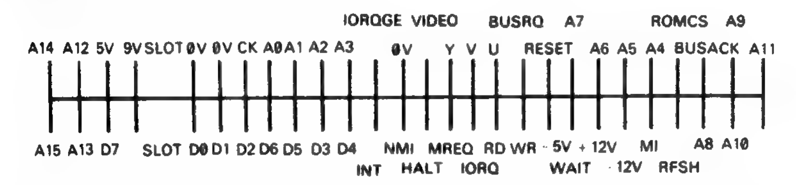
Otro lugar en el que puede colocar datos temporalmente es en el área del archivo de imagen. Deje una pequeña ventana en la pantalla para los mensajes, y en el resto coloque tanto la tinta como el papel del mismo color. Debe evitar usar CLS y también escribir en el área reservada pues se perderían los datos.

CAPÍTULO DUODÉCIMO

Accesorios

COMPLEMENTOS DE HARDWARE

El Spectrum tiene un panel en la parte trasera que le permite usar la mayoría de las líneas del Z80A. Tenga en cuenta que el Spectrum usa todo el mapa de memoria en su versión de 48K y que si añade dispositivos de memoria mapeados al de 16K puede causarle problemas si luego quiere poner la ampliación de memoria. Aquí tiene un diagrama del puerto:



Como puede ver trae las líneas de —12, +12, —5, y +5 voltios para que sea fácil la implementación de un interface RS232. También están las salidas necesarias para un monitor de vídeo. Para ello necesita unir con un cable la salida VID y el modulador de UHF. También necesitará resistencias de 300 Ohm en serie. Desgraciadamente, Sinclair ha olvidado traer la línea RAMCS, así que no será fácil añadir dispositivos de memoria que cierren la memoria interna cuando se desee. Yo pienso que una línea RAMCS puede ser conectada, ya que la posición que ocupa en el ZX81 (si está familiarizado con él) está vacía (corresponde al conector que está al lado del D7). Si usted sólo tiene 16K de RAM y desea utilizar alguno de los muchos dispositivos que se han creado para ampliar la memoria del ZX81 para

la región del Spectrum comprendida entre los 32 y 64K, puede hacerlo identificando las líneas y colocando un conector intermedio que las reordene como las del ZX81.

También se puede colocar una impresora del tipo Centronics. Sólo necesitaría usar las líneas de la 0 a la 6 para los datos, y la línea 7 como BUSY y STROBE. Esto se puede hacer fácilmente por software, aunque es mejor en código máquina, el principio en BASIC sería:

```
100 POKE xxxx,128  strobe (xxxx es dirección de puerto)
110 POKE xxxx,n    donde n es 128 + datos
120 POKE xxxx,128  strobe
130 IF PEEK xxxx = 128 THEN GOTO 130   ¿todavía
140 etc . . . . .                      ocupado?
```

CONVERSACIÓN ENTRE BASICS

Aunque la mayoría de microordenadores trabajan en BASIC, estos BASICS entre sí son un poco diferentes. Otros dos BASICS conocidos son el del ZX81 y el BASIC de Microsoft.

EL BASIC DEL ZX81

El ZX81 tiene un BASIC muy parecido al del Spectrum, hasta tiene también los comandos con una sola tecla, pero hay diferencias en cuanto a la gestión de la pantalla y en el hecho de que el ZX81 no dispone ni de color ni de sonido sin un equipo especial añadido. El Spectrum dispone de alta resolución, mientras que el ZX81 no, ya que el punto más pequeño corresponde a una cuarta parte del cuadrado de un carácter. Así, aunque el comando PLOT de ambas máquinas realice una función similar, da unos resultados muy distintos. En el ZX81 se dispone de una matriz de 64 × 44 mientras que en el Spectrum ésta es de 256 × 176. Quizá la mayor diferencia es que el ZX81 dispone de UNPLOT mientras que con el Spectrum usted debe hacer PLOT con OVER 1.

En otros aspectos, los dos BASICS son idénticos, aunque el Spectrum ofrece más comandos y funciones. También hay pequeños cambios, ya que en el ZX81 CLEAR simplemente borra las variables, y en el Spectrum también coloca

a RAM TOP en la dirección indicada. Por último, diré que el ZX81 usa "**" para las potencias cuando el Spectrum utiliza " ↑ ".

OTROS BASICS (CON ESPECIAL REFERENCIA AL DE MICROSOFT)

Conjuntos

En otros ordenadores, DIM X$ (8) dimensiona un conjunto de 9 elementos, no ocho como en el Spectrum, porque se cuenta también el elemento Ø. En algunos ordenadores, se permite dimensionar varios conjuntos en una sola sentencia DIM. En el Spectrum DIM A$ (6), B$ (4) debe sustituirse por DIM A$ (6): DIM B$ (4).

GET, GET$, INKEY, INKEY$

En el Spectrum sólo tiene la opción de usar INKEY$ que devuelve la tecla que se está pulsando en aquel momento. Muchos otros BASICS también disponen de esto, pero algunos requieren que se añada detrás un número entre paréntesis que indique por cuánto tiempo se deben esperar. Para hacer esto con el Spectrum tendría que utilizar un bucle FOR NEXT alrededor de INKEY$.

Algunos BASICS, también disponen de INKEY, que devuelve el código ASCII de la tecla pulsada. GET lo tienen otros ordenadores, y es similar a INKEY. Sin embargo, GET se espera indefinidamente. Finalmente GET$ es como INKEY$ pero también espera indefinidamente. El INKEY$ del Spectrum también puede simular el GET$ pues para obtener el código de ASCII sólo tiene que hacer: LET A = CODE INKEY$.

GOTO Y GOSUB

Algunos ordenadores, no permiten poner una expresión detrás de GOTO sino únicamente un número.

IF-THEN, IF-THEN-ELSE

Algunos ordenadores le permiten escribir frases como:

IF A$ = "YES" THEN 400 ELSE...

El Spectrum le obliga a poner GOTO antes del número y no dispone de ELSE que es lo que procede a la sentencia que se ejecutará en caso de que la condición no se cumpla. Otros ordenadores poseen diferente lógica y no permiten algo que hemos usado mucho a lo largo del libreo:

10 IF A THEN GOTO 100

ON-GOTO, ON-GOSUB

Estas dos instrucciones de las cuales carece el Spectrum trabajan del siguiente modo:

```
10 INPUT X
20 ON X GOTO 200,300, 456, 9998
```

Esto provocaría que el programa saltara a la línea 200 si se introduce un 1, a la 300 si es un 2, a la 456 si se entra un 3 y a la 9998 si lo que se introduce es un 4. Simular esto con el Spectrum no es difícil, basta emplear un uso directo de AND:

```
10 INPUT A
20 GOTO (200 AND A = 2) + (456
AND A = 3) + (9998 AND A = 4)
```

También puede hacerse dimensionando un conjunto:

```
10 DIM A(4)
20 LET A (1) = 200: LET A(2) = 300: LET A(3) =
456: LET A(4) = 9998

500 INPUT X
510 GOTO A(X)
```

De un modo similar se puede reproducir ON-GOSUB

Until

Until significa hasta. Algunos ordenadores le permiten repetir un proceso hasta que se cumple una cierta condi-

ción. Esto se puede copiar en el Spectrum usando un GOTO y un IF-THEN-GOTO que se salga del bucle si se detecta el cumplimiento de la condición:

```
10 LET A = 0
20 LET A = A + 1
30 IF A<>20 THEN GOTO 20
or
10 IF INKEY$<>"S" THEN GOTO 10
```

PROC, END PROC

Un procedimiento es una subrutina más sofisticada a la que se le pueden pasar variables y que puede ser llamada en cualquier momento. Ya vimos el modo de simular la llamada (asignando a una variable el número de línea en que empieza la subrutina). Pero lo de pasar variables ya es un poco más complicado.

LET

Algunos BASICS permiten omitir la palabra LET, pero en el Spectrum no se puede.

LEFT$, RIGHT$, MID$, ETC

En otros ordenadores, la manipulación de cadenas se hace usando estos comandos. Aquí se muestran sus equivalentes con los del Spectrum:

```
LEFT$ (A$,3)      A$ (TO 3)
RIGHT$ (A$,4)     A$[(LENA$ - 4) TO LEN A$]
MID$ (A$,1,3,)    A$(1 TO 3)
```

INSTR$ también existe en algunos ordenadores, y tiene la forma:

INSTR$ (A$, B$)

y busca la presencia de la segunda cadena en el interior de la primera. En el Spectrum se puede substituir por esta rutina:

```
10 FOR A = 1 TO LEN A$ - LEN B$
20 IF A$(A TO A + LEN B$) = B$THEN PRINT
   "YES"
30 NEXT A
```

Matemáticas

Algunos BASICS permiten definir números enteros y de coma flotante; % significa entero. DIV da el número entero resultado de la división:

PRINT 13 DIV 4

da como resultado 3 (olvidando el resto de 1). Este resto, se encuentra usando MOD:

PRINT 13 MOD 4

da como resultado 1.

Algunos BASICS también pueden convertir números entre decimal y hexadecimal usando HEX o H. Además un número se puede definir directamente en hexadecimal poniendo previamente HEX o H. Así HA4 indica que 44 es un número hexadecimal.

PLOT

En muchos ordenadores en lugar de PLOT hay que usar SET y en lugar de PLOT OVER hay que usar RESET. Para la alta resolución existen dos palabras más, que son PSET y PRESET. En algunos ordenadores, se usa LINE en lugar de DRAW, del siguiente modo:

LINE (5,3) — (6,6), PSET

Esto dibuja una línea desde el punto (5,3) hasta el punto (6,6). En el Spectrum hay que calcular el punto final a partir del inicial usando una combinación de PLOT y DRAW.

Almacenamiento en cinta

Muchos ordenadores tienen CLOAD y CSAVE en lugar

de LOAD y SAVE. También pueden usar CHAIN en lugar de MERGE.

PRINT

Algunos BASICS usan PRINT TAB (A × B) donde nosotros usamos PRINT AT (A, B). Otros utilizan PRINT @ seguido de otro número que le indica al ordenador una posición en la pantalla. Es como el PRINT AT del Spectrum sólo que condensa los números de fila y columna en uno solo. Finalmente, algunos BASICS permiten usar '?' en lugar de PRINT.

APENDICES

I TABLA DE CARACTERES ASCII

0	)	36	$	71	G
1	)	37	%	72	H
2	) no usados	38	&	73	I
3	)	39	‘	74	J
4	)	40	(	75	K
5	)	41	)	76	L
6	coma de PRINT	42	*	77	M
7	EDIT	43	+	78	N
8	izquierda	44	,	79	O
9	derecha	45	-	80	P
10	abajo	46	.	81	Q
11	arriba	47	/	82	R
12	DELETE	48	0	83	S
13	ENTER	49	1	84	T
14	número	50	2	85	U
15	no usado	51	3	86	V
16	INK control	52	4	87	W
17	PAPER control	53	5	88	X
18	FLASH control	54	6	89	Y
19	BRIGHT control	55	7	90	Z
20	INVERSE control	56	8	91	[
21	OVER control	57	9	92	\
22	AT control	58	:	93	]
23	TAB control	59	;	94	↑
24	)	60	<	95	—
25	)	61	=	96	£
26	)	62	>	97	a
27	) no usados	63	?	98	b
28	)	64	@	99	c
29	)	65	A	100	d
30	)	66	B	101	e
31	)	67	C	102	f
32	espacio	68	D	103	g
33	!	69	E	104	h
34	‘‘	70	F		
35	#				

105	i	125	}	145	(b))	
106	j	126	-	146	(c))	
107	k	127	©	147	(d))	
108	l	128	□	148	(e))	
109	m	129	▝	149	(f))	
110	n	130	▘	150	(g))	
111	o	131	▀	151	(h))	
112	p	132	▗	152	(i))	
113	q	133	▐	153	(j))	gráficos
114	r	134	▚	154	(k))	de
115	s	135	▜	155	(l))	usuario
116	t	136	▖	156	(m))	
117	u	137	▞	157	(n))	
118	v	138	▌	158	(o))	
119	w	139	▛	159	(p))	
120	x	140	▄	160	(q))	
121	y	141	▟	161	(r))	
122	z	142	▙	162	(s))	
123	{	143	█	163	(t))	
124	|	144	(a))	164	(u))	

II MAPA DE MEMORIA

Zona	Dirección / Nota
	65536 – Ultima posición de memoria
GRAFICOS DEFINIBLES	
	'USR"a"' (Depende del modelo)
	← RAMTOP (Variable)
AREA DE PROGRAMA BASIC	
	23734
VARIABLES DEL SISTEMA	
	23552
BUFFER IMPRESORA	PRINTER BUFFER
	23296
FICHERO DE ATRIBUTOS	
	22528
PANTALLA	
	16384
R.O.M. MEMORIA DE SOLO LECTURA	(MONITOR, BASIC, ETC).

ÍNDICE

Este libro del Spectrum es el mejor valor en libros (Spectrum)... contiene un amplio número de consejos y muchas utilidades ... Especialmente para quienes estén interesados en gráficos

"Practical Computing"; enero 1983

Este libro del Spectrum *ofrece a sus lectores la oportunidad de acoplarse a la potencia del más excitante microordenador de la década el ZX-Spectrum. El Dr. Tim Langdell cree que el amplio potencial debería estar al alcance de todos los propietarios del Spectrum-justamente de aquellos que no tienen experiencia previa en computadoras. Su introducción paso a paso, abre el mundo del Spectrum a los principiantes, mientras los enriquecedores consejos y sugerencias, excitantes programas y comprensivos capítulos dedicados al código-máquina y aplicaciones, hacen al libro una lectura esencial para los programadores expertos.*
Este libro del Spectrum *debe ser para cualquiera el medio de llevar a su Spectrum hasta el límite y más allá.*

EDITORIAL NORAY, S.A.
San Gervasio de Cassolas, 79
Barcelona